Recent HEP.TrkX results

Xiangyang Ju, Steve Farrell

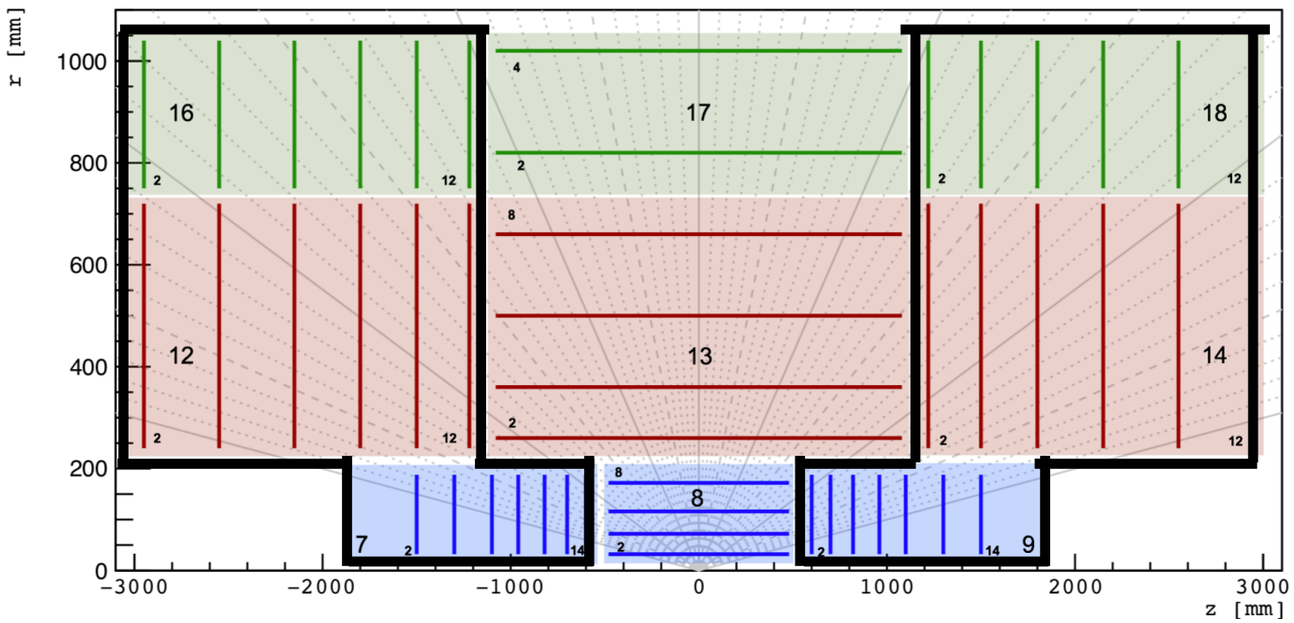
Exa-TrkX kick-off meeting, 4 June 2019

Introduction

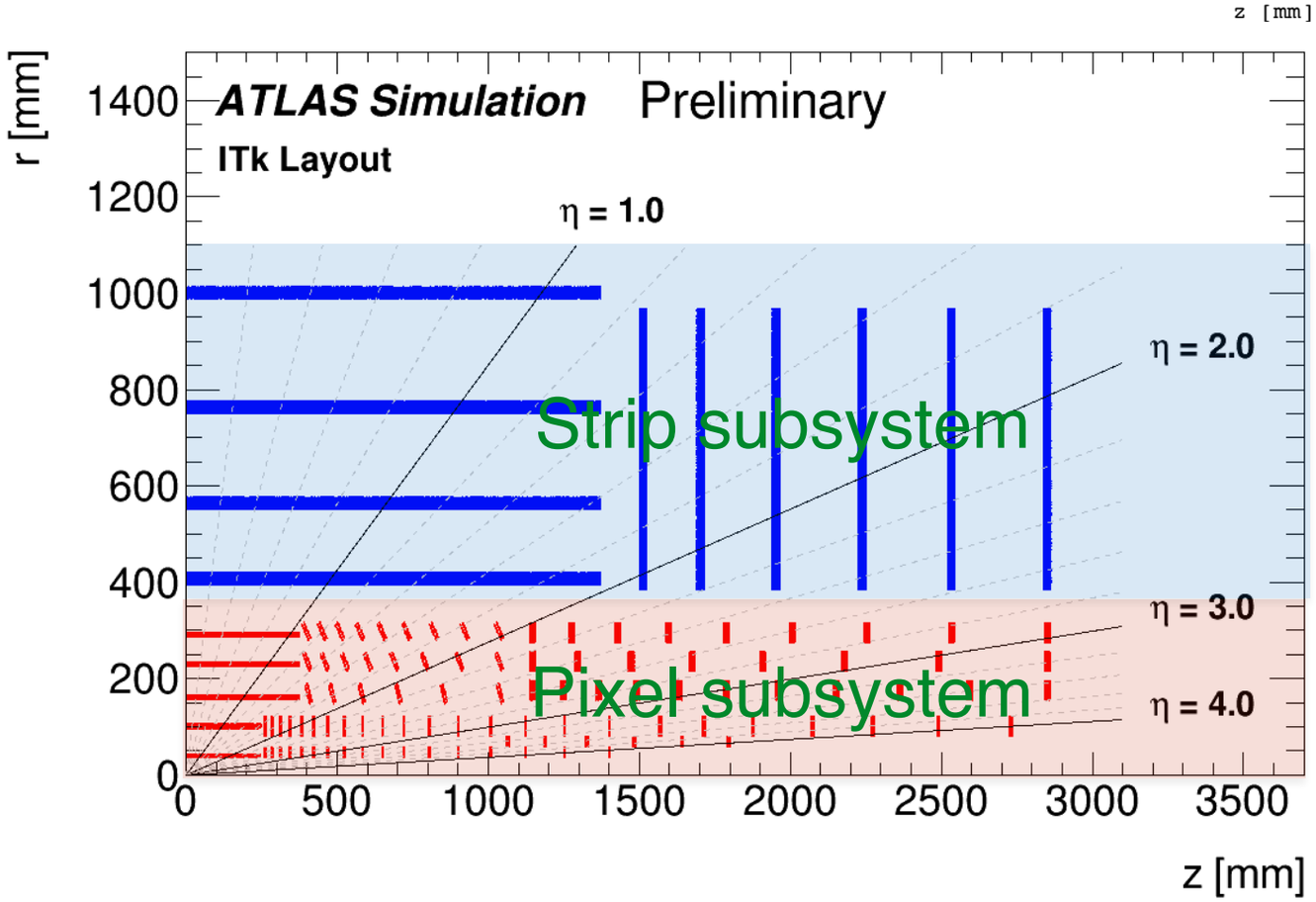
1. Recap the results presented at the 2019 Connecting The Dots
2. Recent results after that
3. Future plans

Tracking ML challenge data

[link to the website]



The data provides simulated hit positions in the inner detector with the geometry shown left, serving as a starting point, but not representing reality.

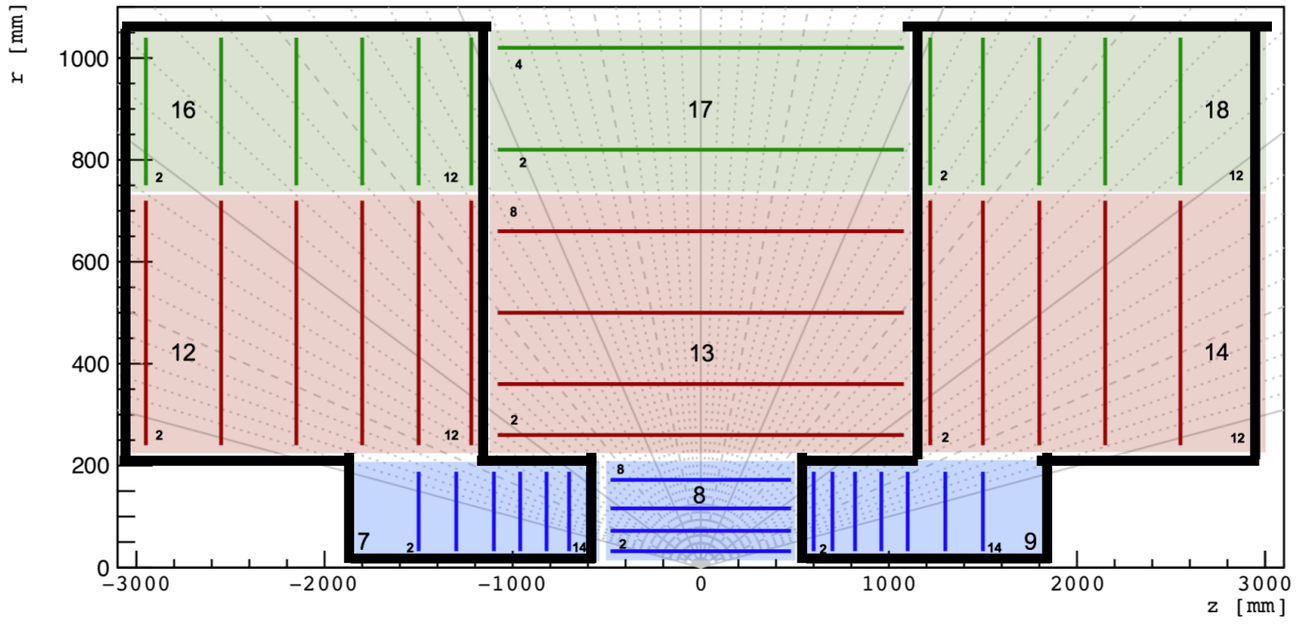


Latest layout of ATLAS Inner Tracker (ITk) ATL-PHYS-PUB-2019-014

Sophisticated layout in pixel endcap.

Start from small

[Link to the talk]



The data is to represent the HL-LHC conditions with μ = 200.

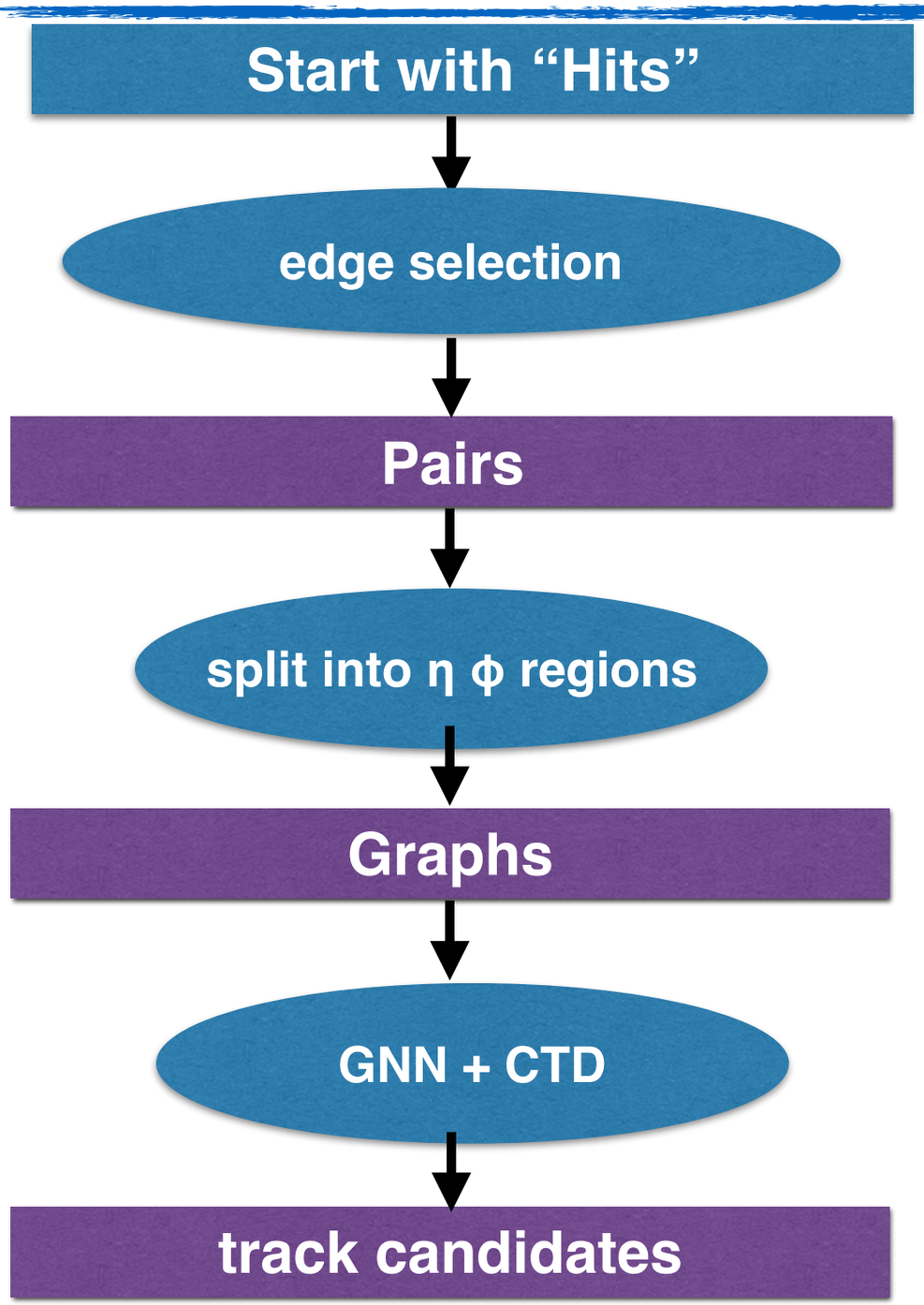
One event has about 10k particles and 100k hits.

Out of the 10k particles, about 86% are **reconstructable**, defined as particles leaving hits in the detector

Tackling the problem from a smaller dataset:

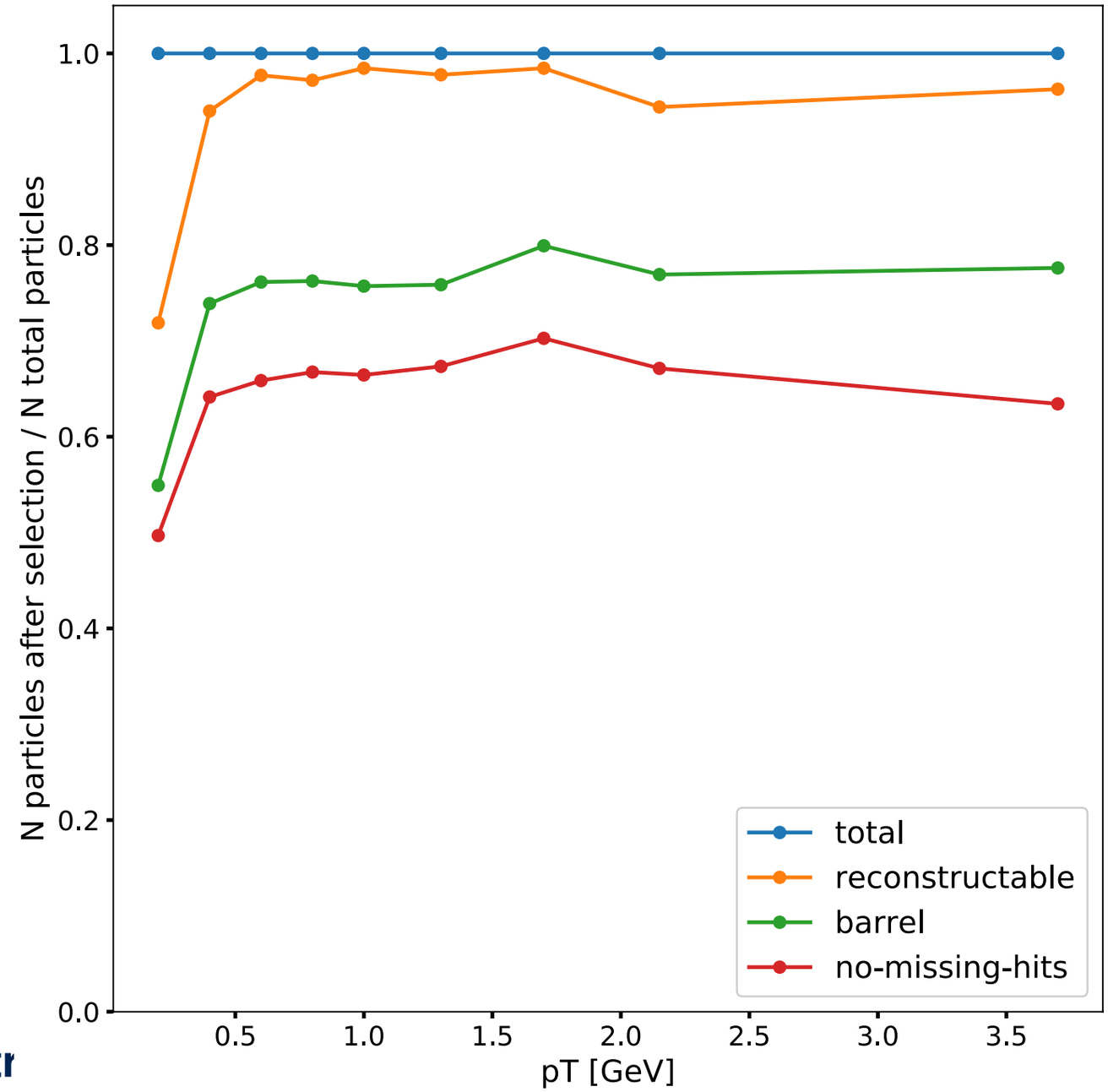
- Hits recorded in volume [8, 13, 17], basically barrel region
- Particles leaving fully connected tracks in the detector, i.e. no missing hits

Track formation

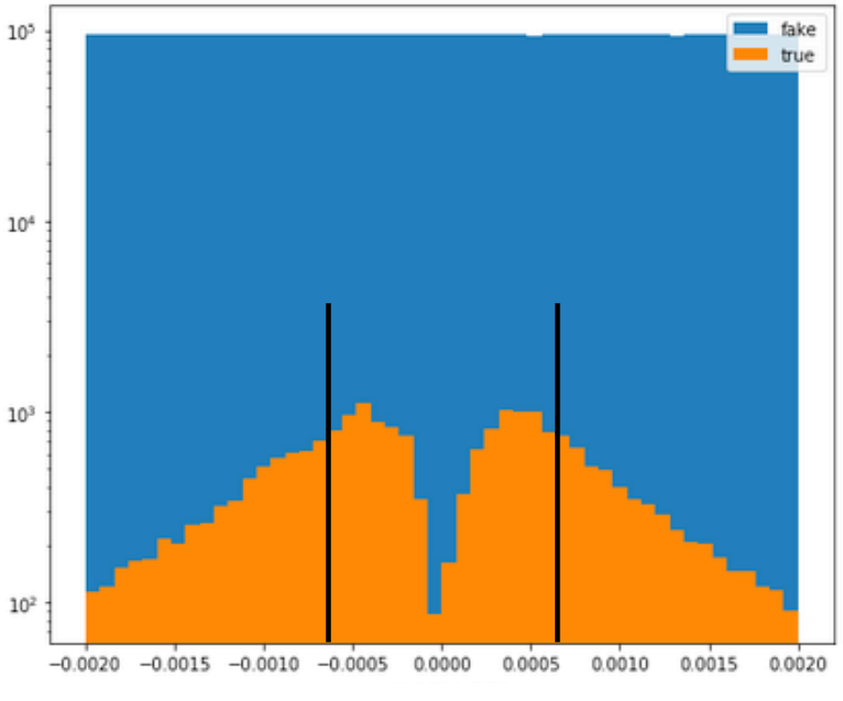


Track candidates survive selections only when they have the exact hits as true tr

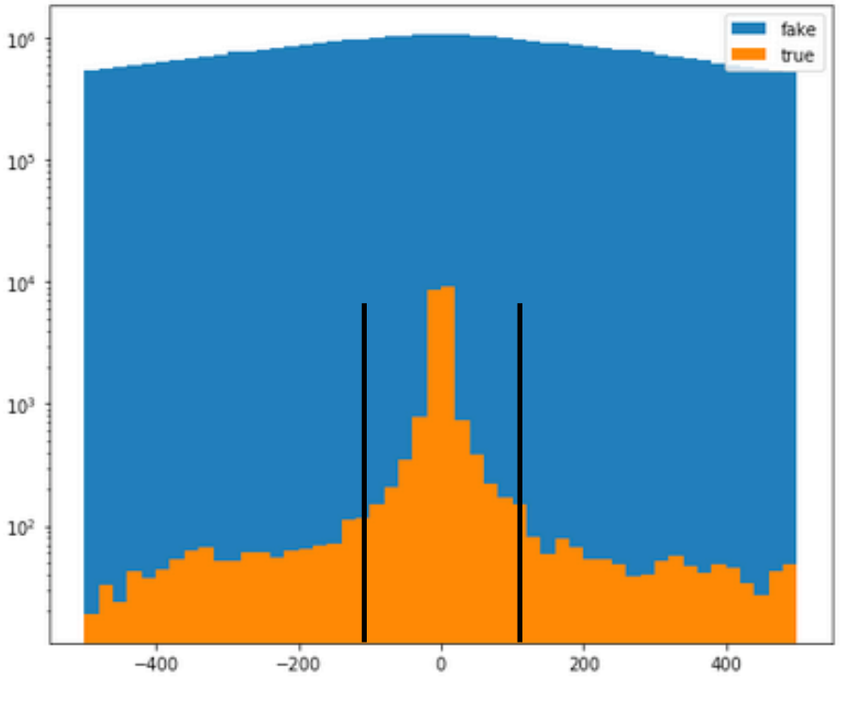
# of particles after stacked selections over the total # of particles in the event



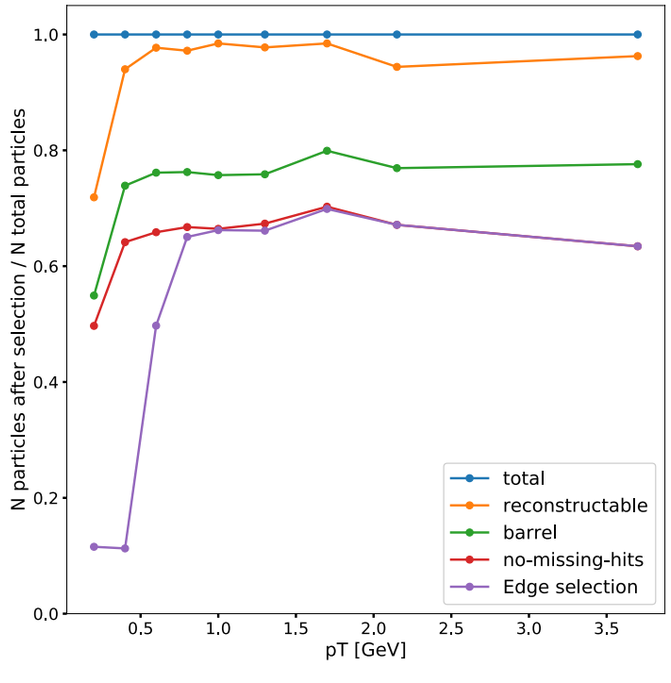
Graph Formation: edge selection



$\Delta\phi/\Delta r$ [rad/mm]

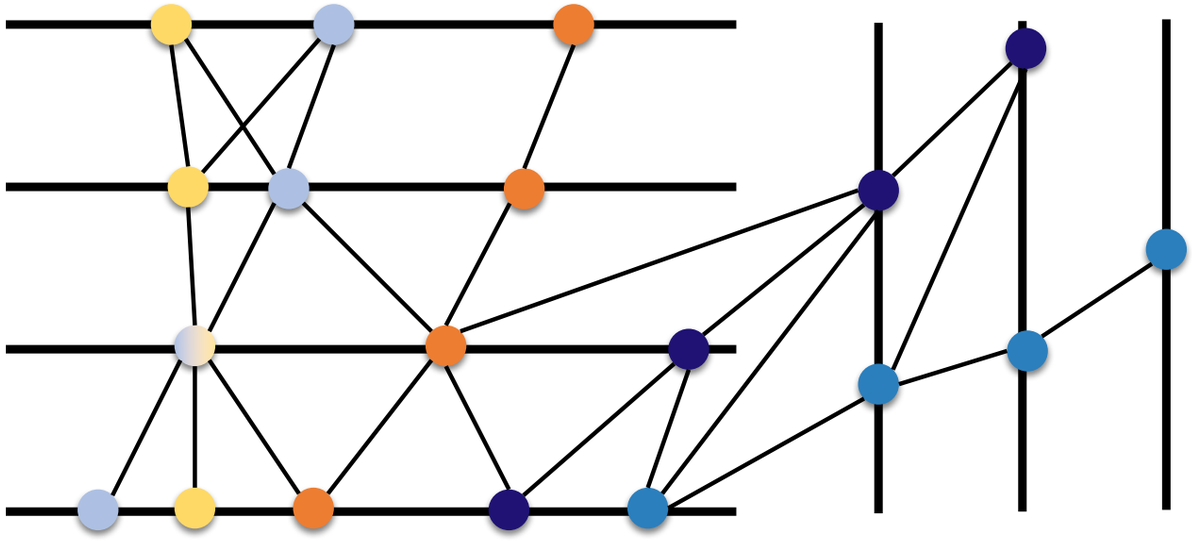


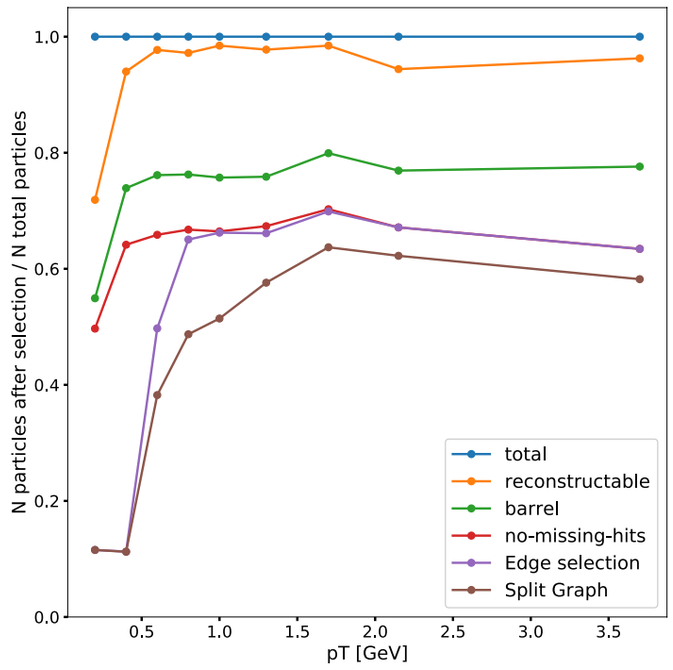
z_0 [mm]



- Make initial edges from hits in adjacent layers
- Use a simple selection to prune away fake edges
  - $\Delta\phi/\Delta r < 0.0006$,
  - z_0 (intercept of the line passing through the two hits) < 100 mm
- Tuned to be efficient for tracks with pT > 1 GeV

Graph Formation: construct graph from pairs

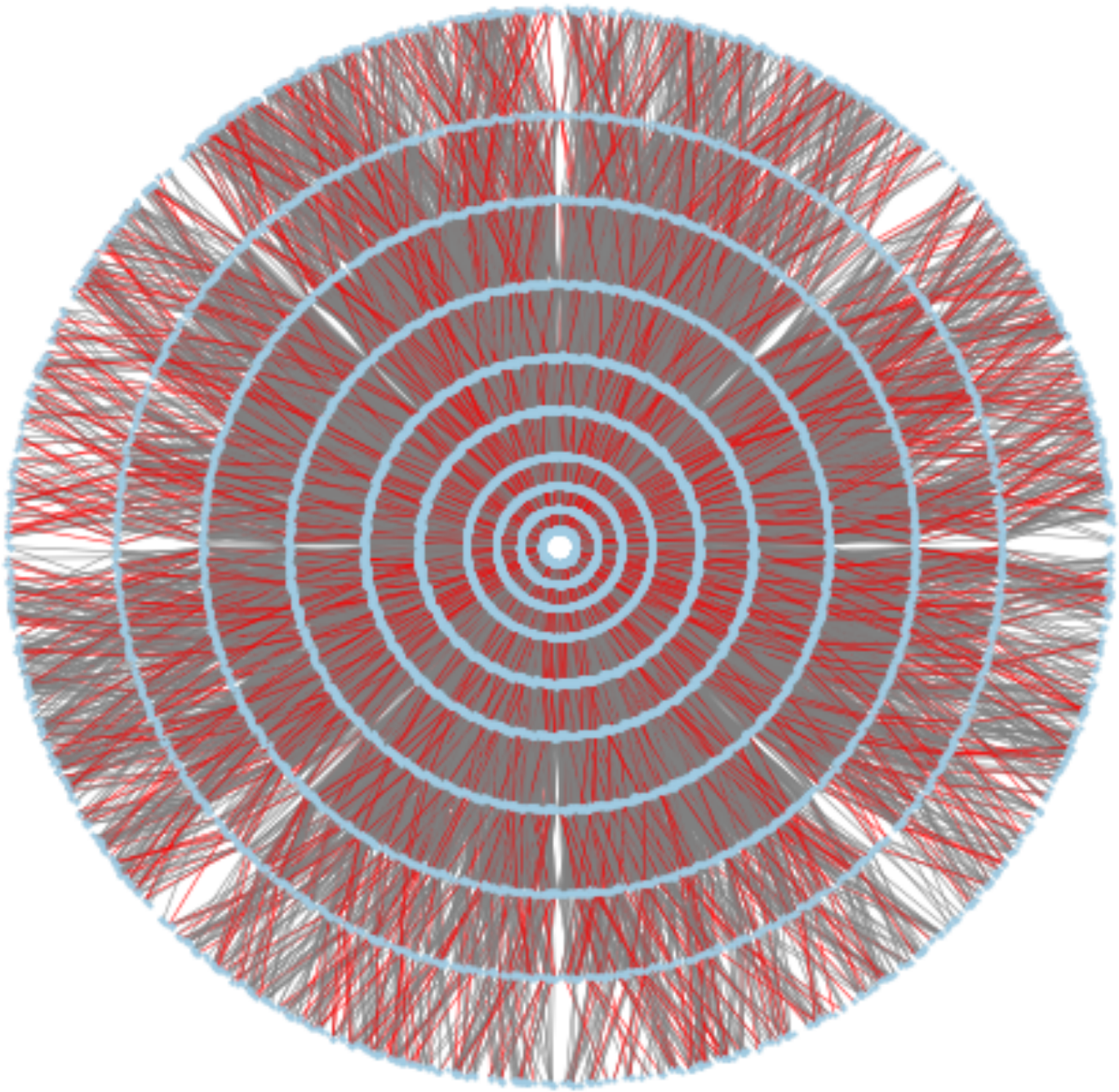


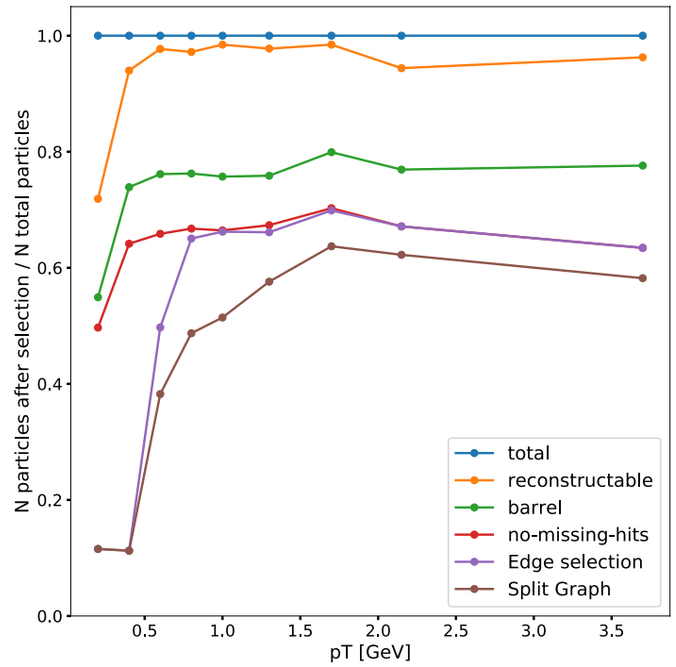


- Construct a directed graph, flowing inside-out
  - Split into 16 subgraphs, **8 ϕ bins and 2 η bins**
    - need smart algorithm to deal with hits in boundaries
  - **input node features**: [r, ϕ, z]
  - Edges belong to a track assigned with score of 1, 0 otherwise

Graph for one event

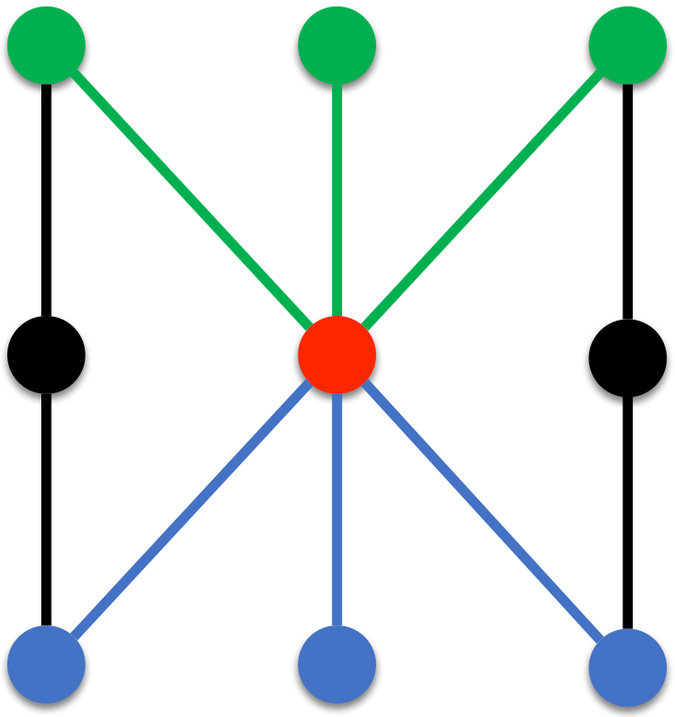
After all previous selections





- About 160k edges, 92% are fake (in gray)
- 8 gaps result from 8 sections in ϕ
- Can GNN find the 13k true edges out of the 147k fake ones?

GNN Architecture



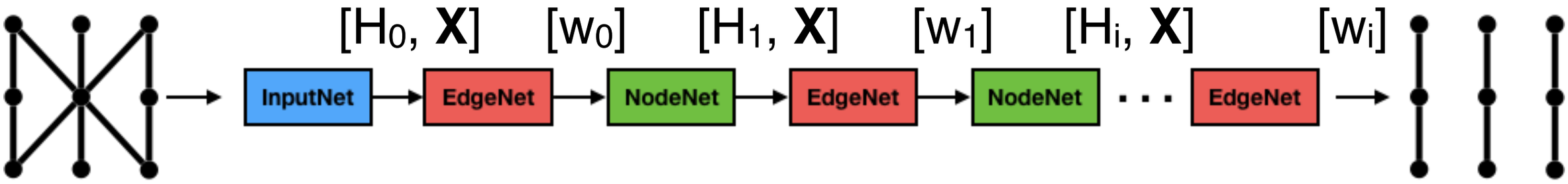
Three components operate on graph:

- **Input network** computes hidden node features
- **Edge network** computes **edge scores** from node features
- **Node network** computes hidden node features from aggregated **weighted incoming** and **outgoing** node features

Incoming and outgoing nodes with higher weights get more “attention”

Putting them together

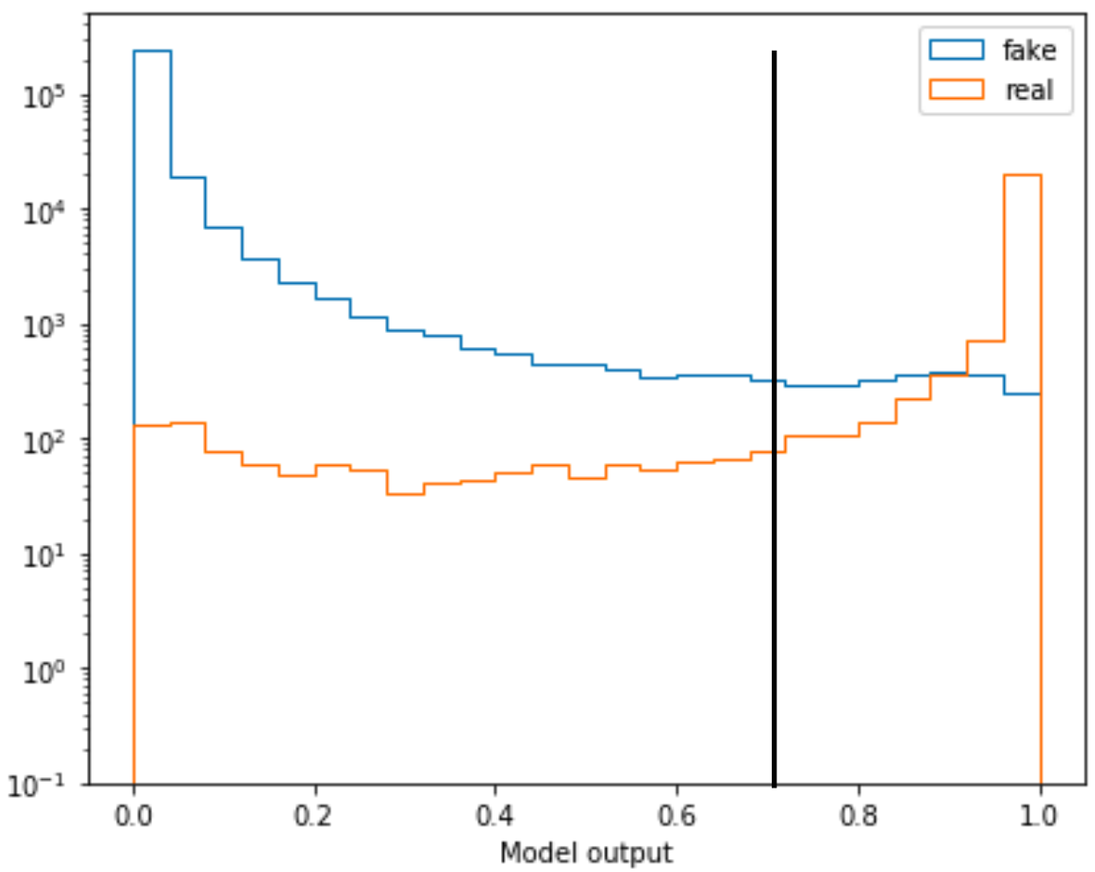
H: hidden states/features, X: input node features



- **EdgeNet and NodeNet iteratively applied 8 times, so 8 iterations**
- Message passed with the “attention mechanism”
- Hidden node features carry embedded track information for Edge Network to make predictions

GNN Output and performance

- ~43k tunable parameters in pytorch
- Trained on NVIDIA V100 'Volta' GPU for about 60 epochs
- Weighted loss function



With a threshold of 0.7:

Edge Efficiency: 95.2%

Edge Purity: 90.2%

$$\text{Efficiency} = \frac{\#\text{ of True Edges passed the threshold}}{\#\text{ of total True Edges}}$$

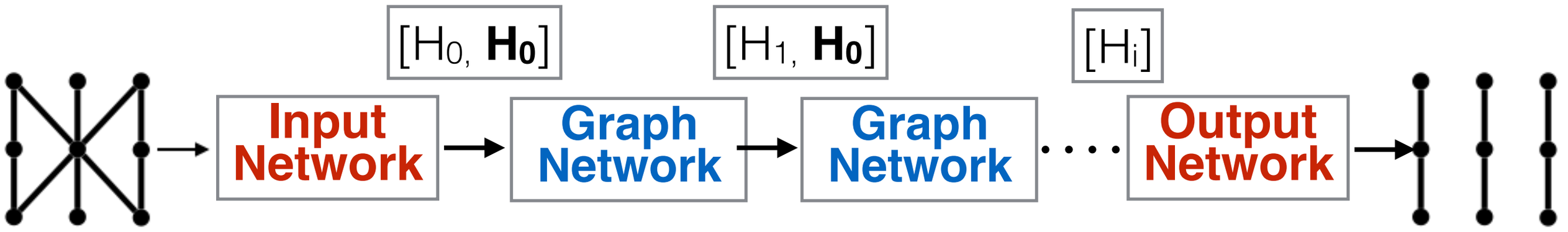
$$\text{Purity} = \frac{\#\text{ of True Edges passed the threshold}}{\#\text{ of total Edges passed the threshold}}$$

- One can use higher threshold to gain purity at the cost of less efficiency.

We are exploring other GNN architectures to push the performance further.

Following example uses graph_nets library from DeepMind and a model resembling the Interaction Networks [link]

Alternative implementation of GNN

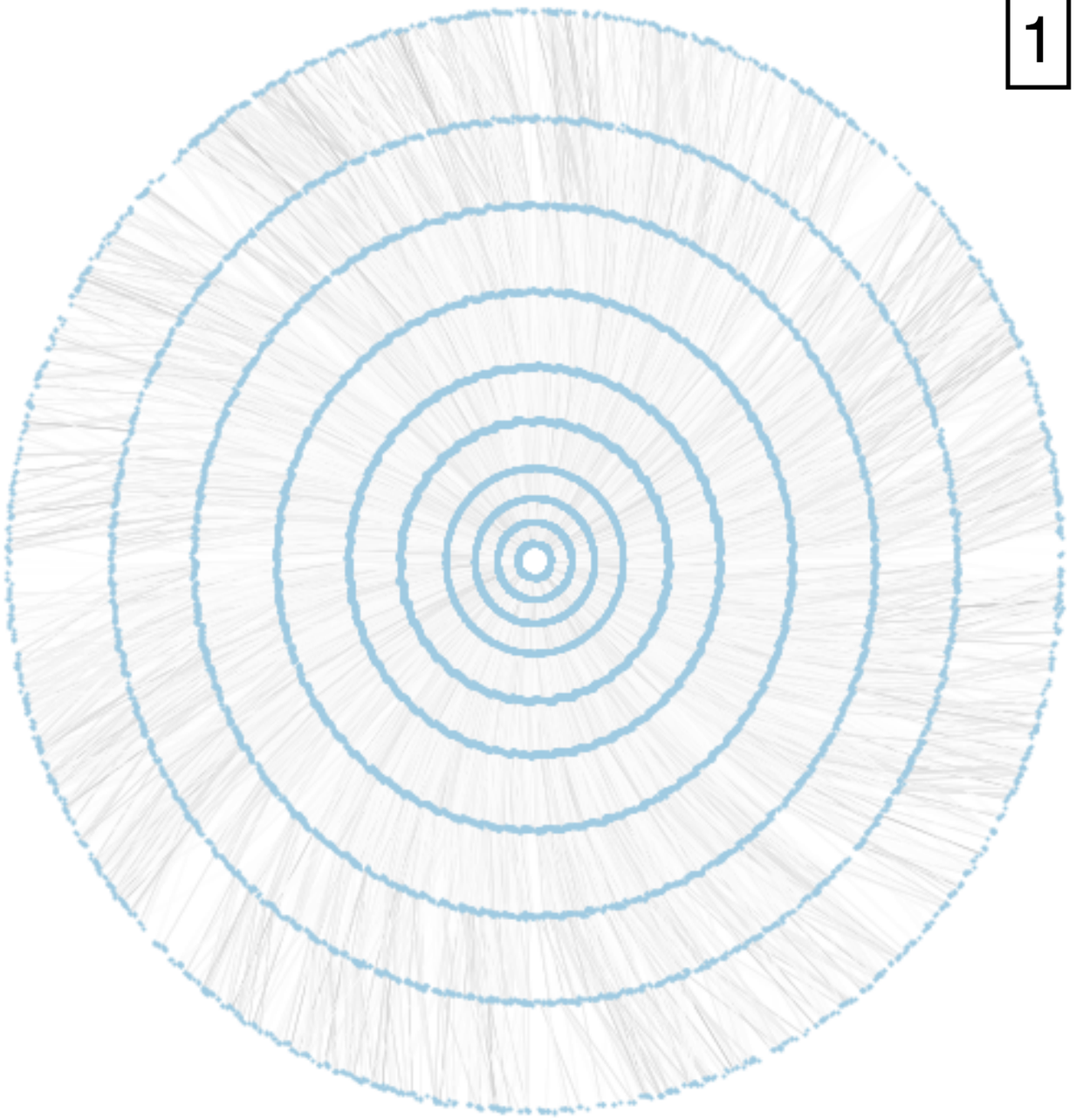


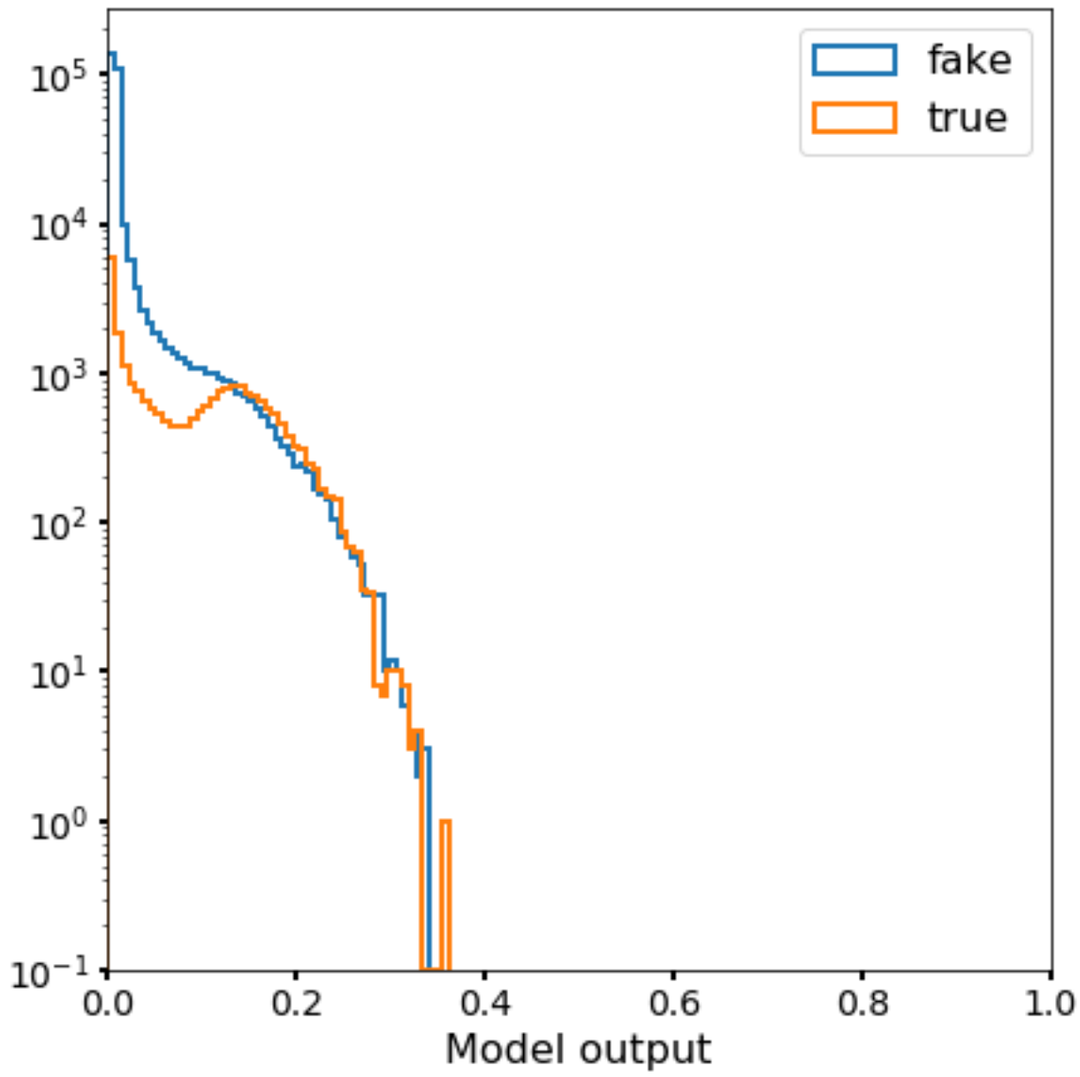
Differences:

- Edge features provided in the input
- Alternate message passing implementation:
  - No explicit attention mechanism
  - Edge features are computed from node features and then summed across all neighbors
- Output Network computes final edge scores
- Bigger, deeper model (266k parameters)

We can visualize the intermediate outputs of the model

Predicted score improves after each iteration

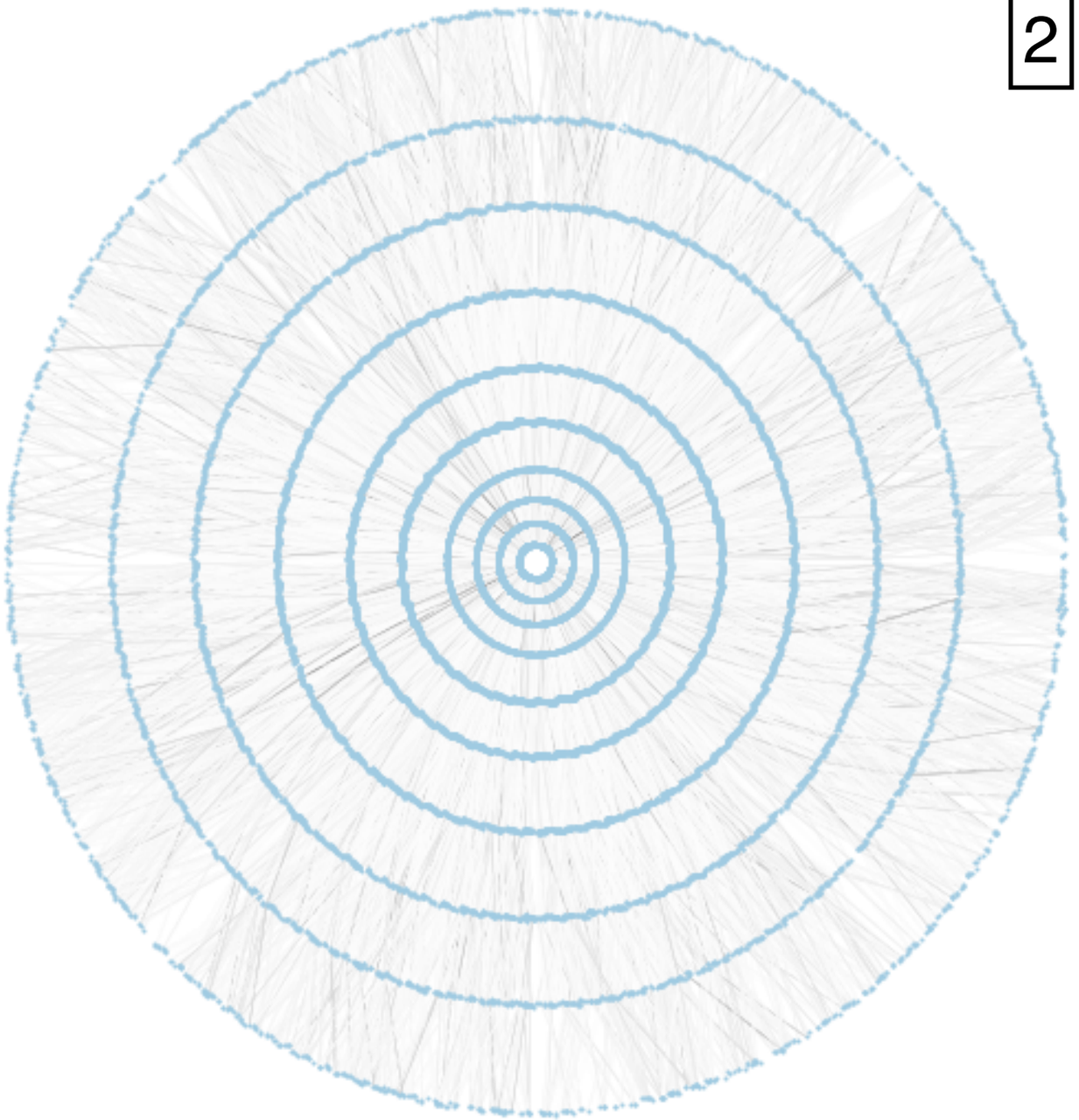


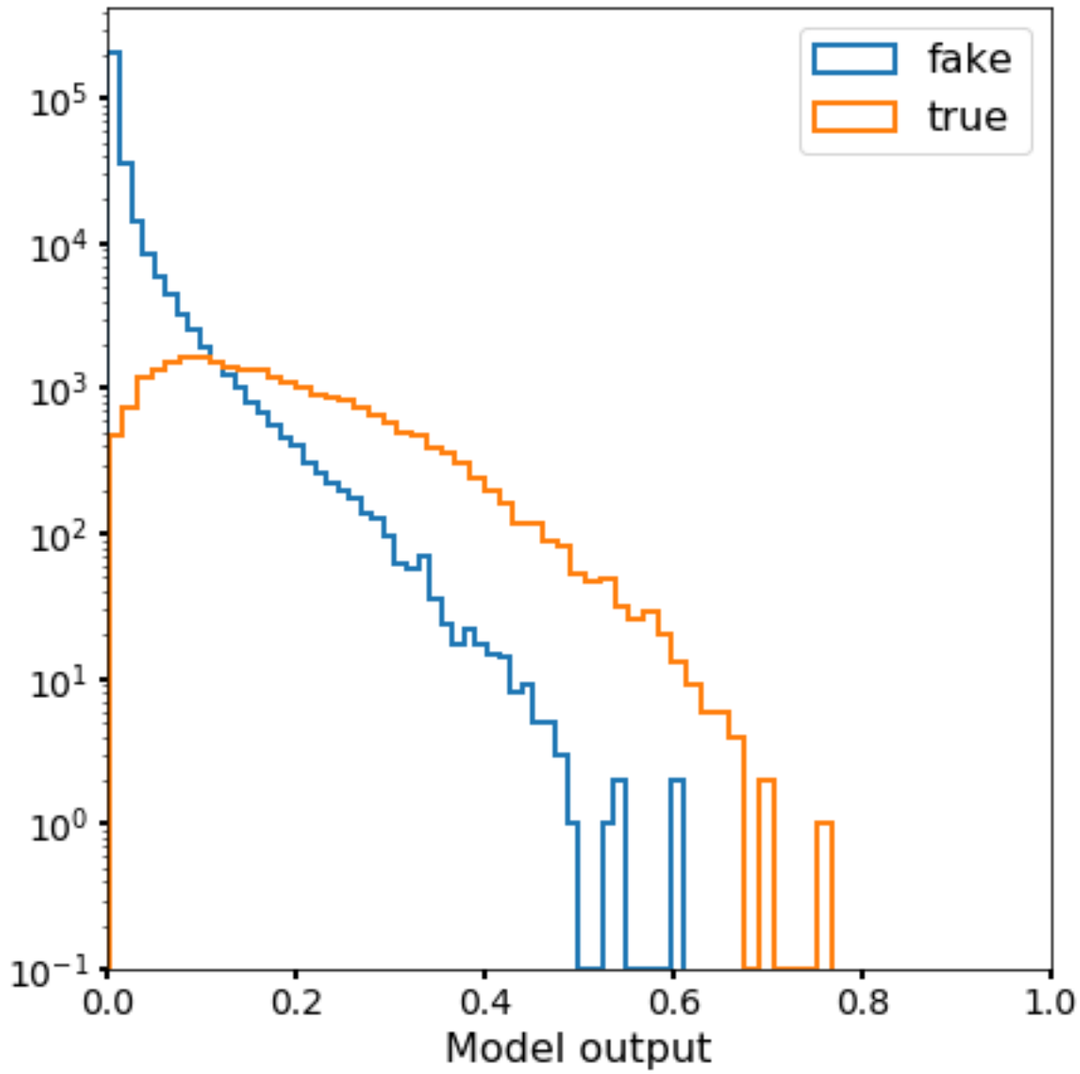


Edges with higher scores are darker than that with lower scores

Edges with scores < 0.01 are removed for visualization purpose.

Predicted score improves after each iteration

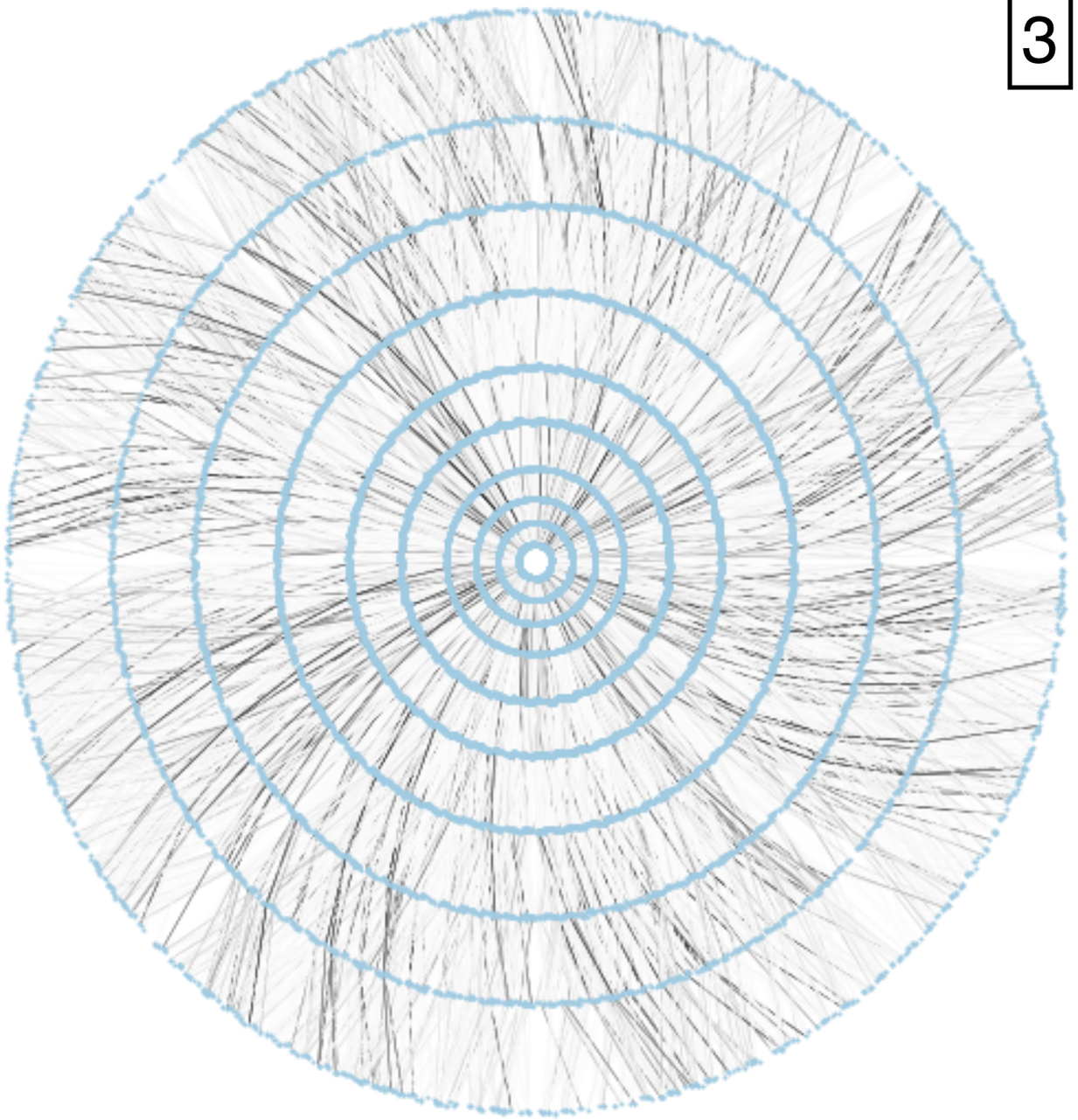


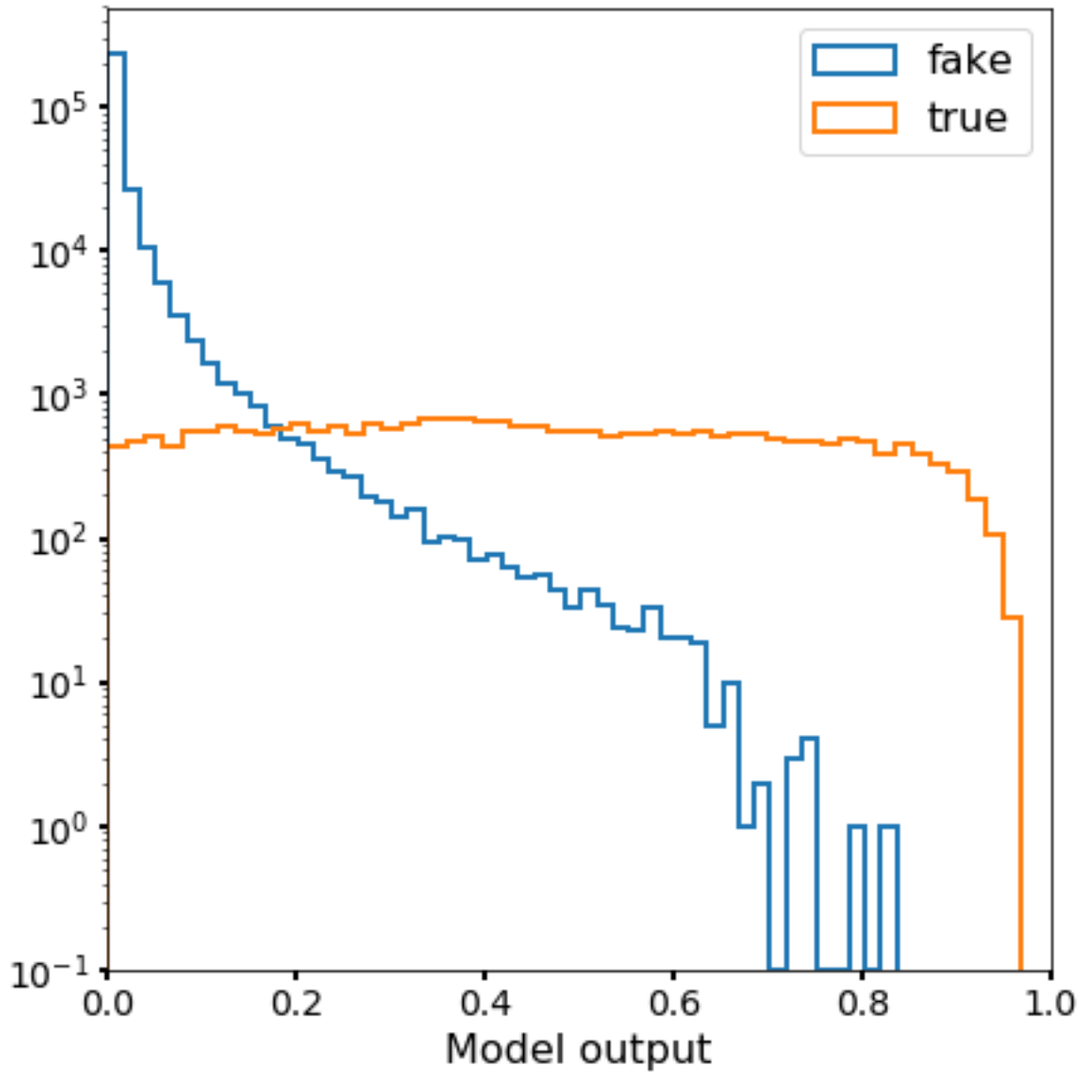


Edges with higher scores are darker than that with lower scores

Edges with scores < 0.01 are removed for visualization purpose.

Predicted score improves after each iteration

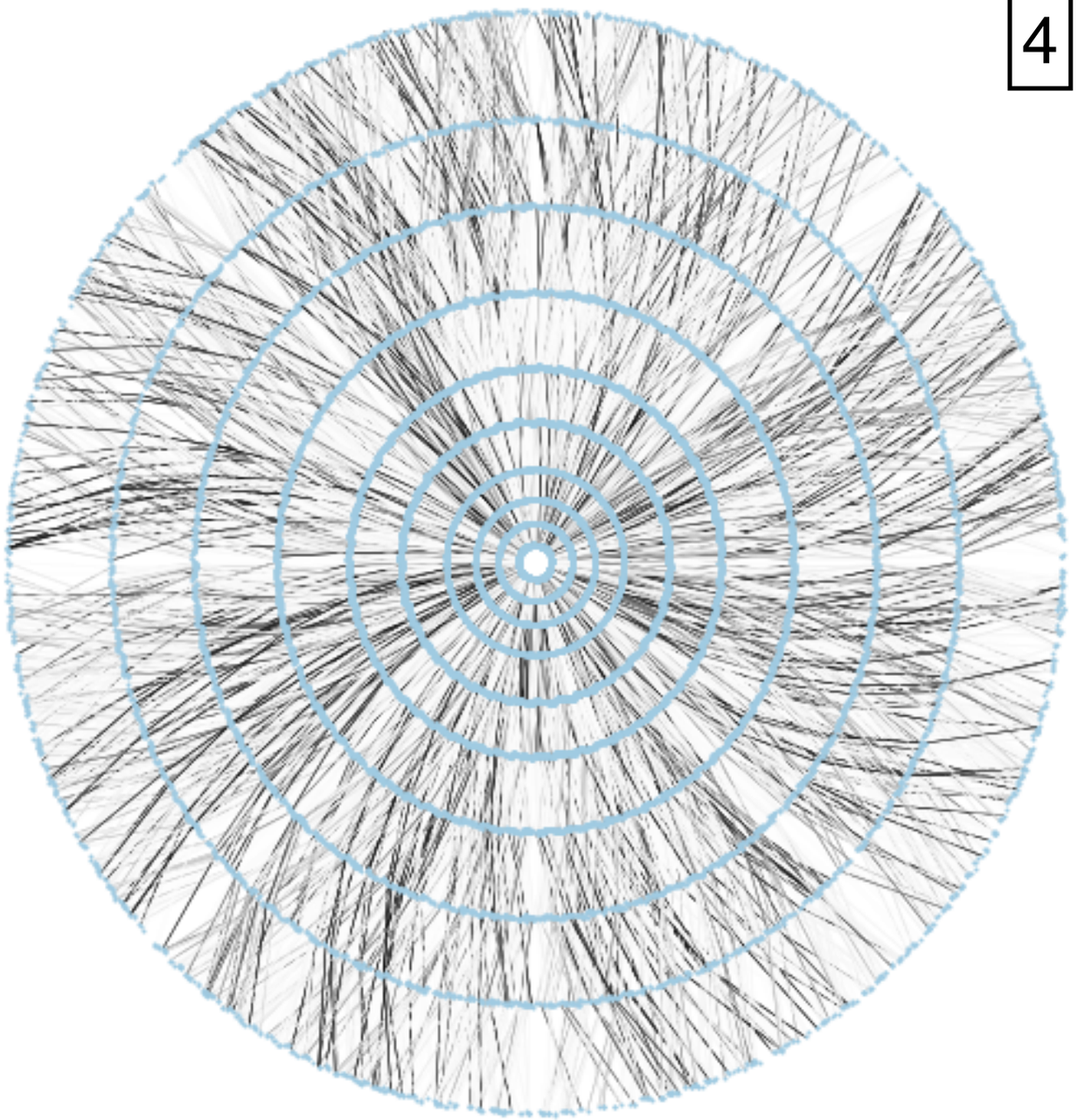


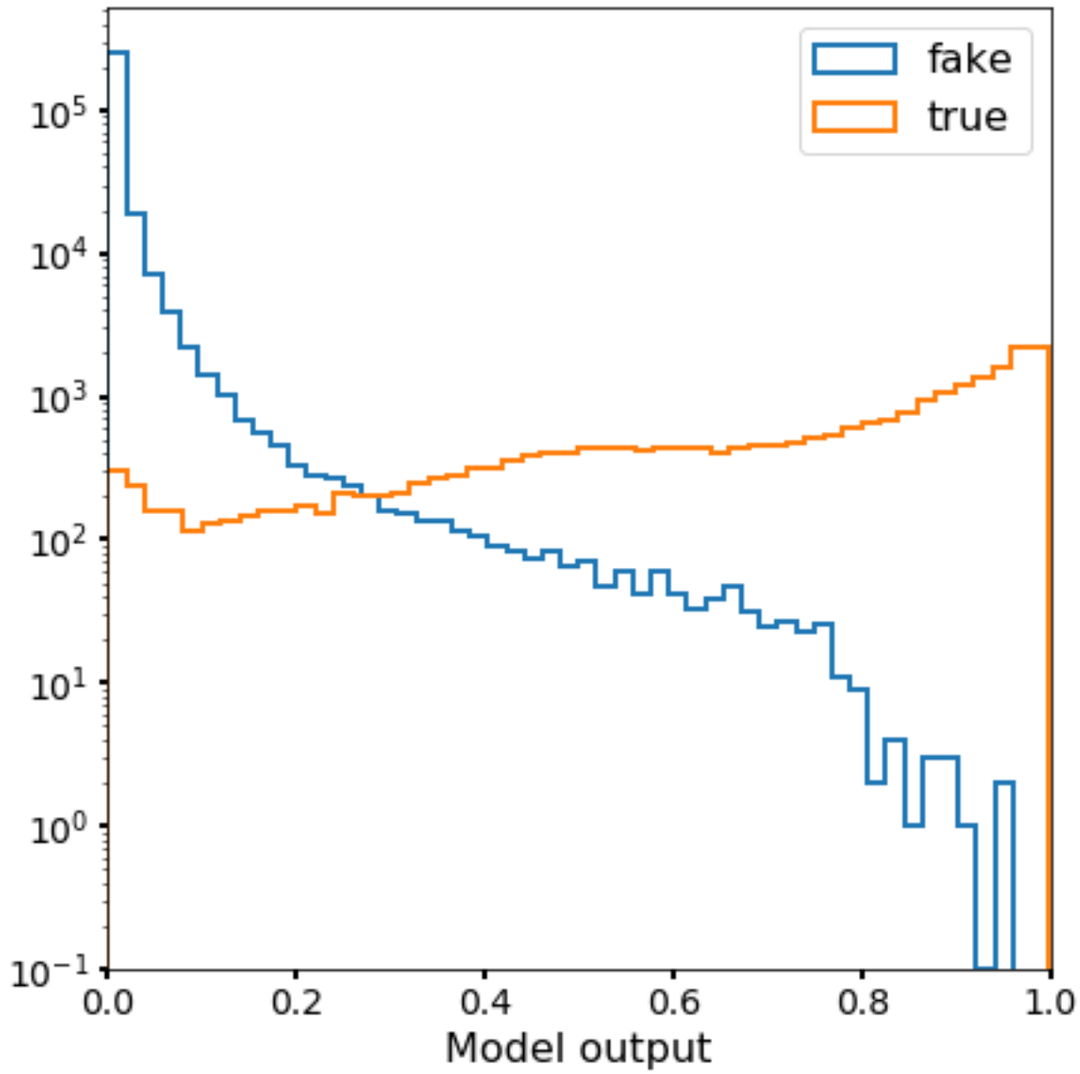


Edges with higher scores are darker than that with lower scores

Edges with scores < 0.01 are removed for visualization purpose.

Predicted score improves after each iteration

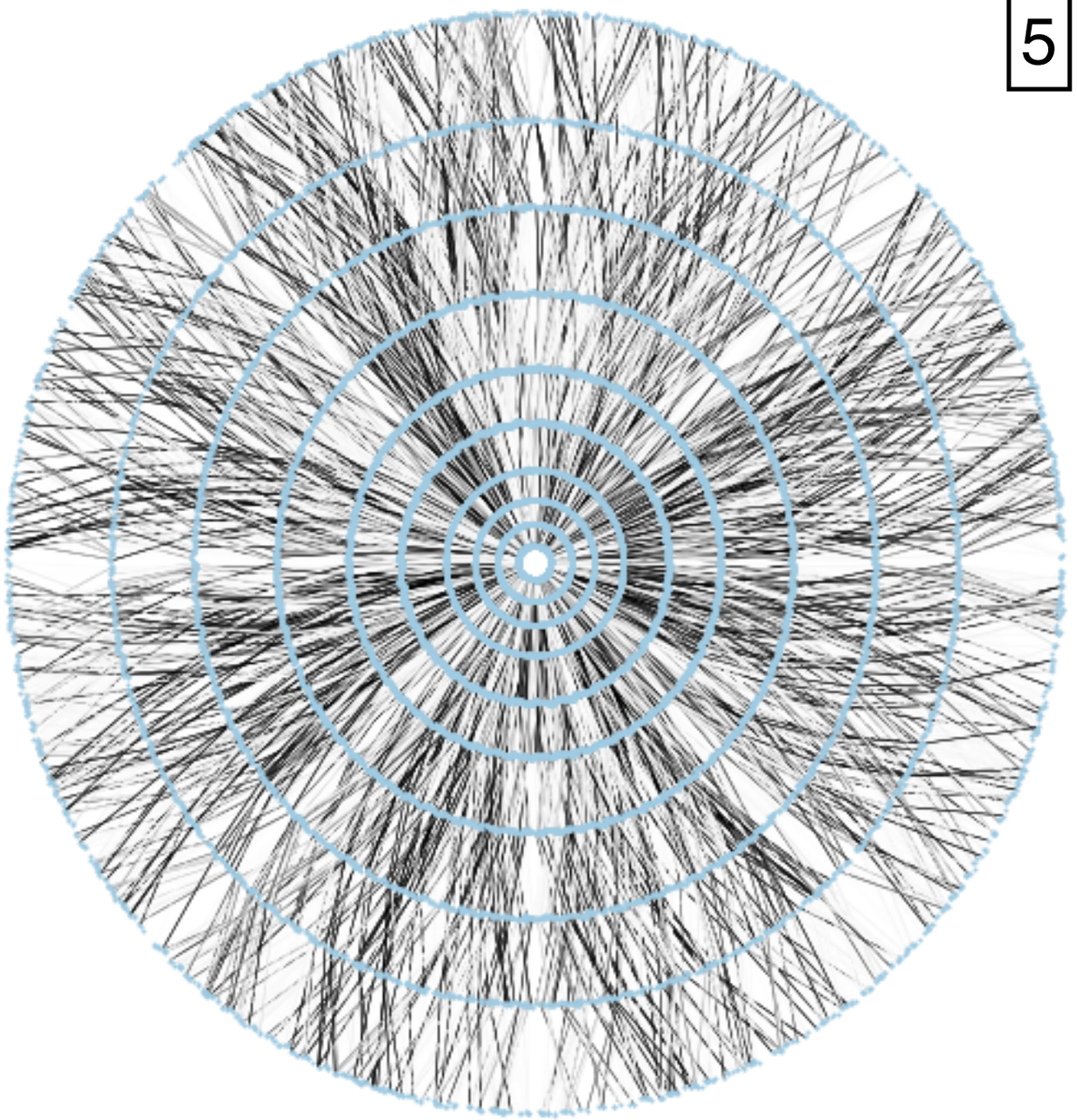


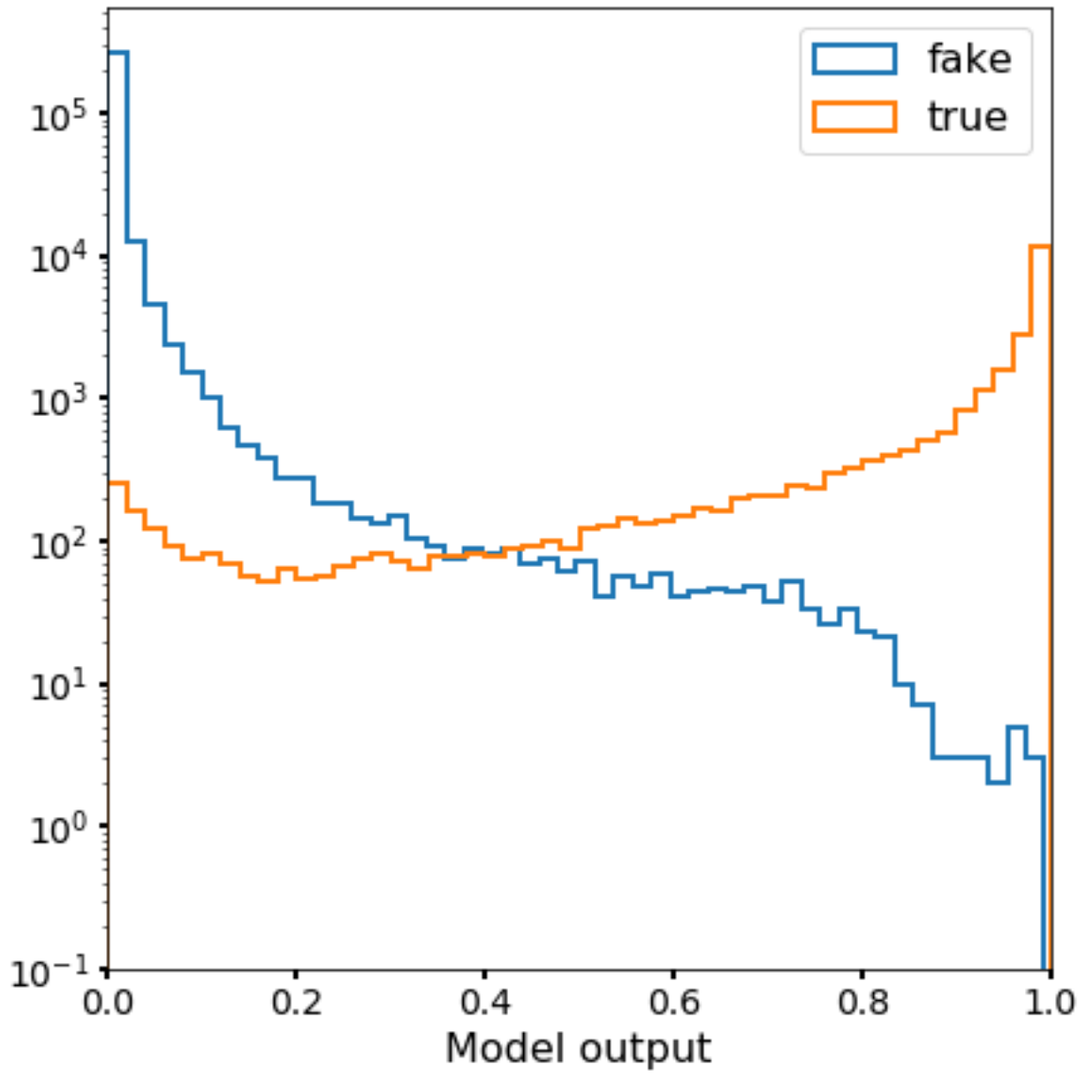


Edges with higher scores are darker than that with lower scores

Edges with scores < 0.01 are removed for visualization purpose.

Predicted score improves after each iteration

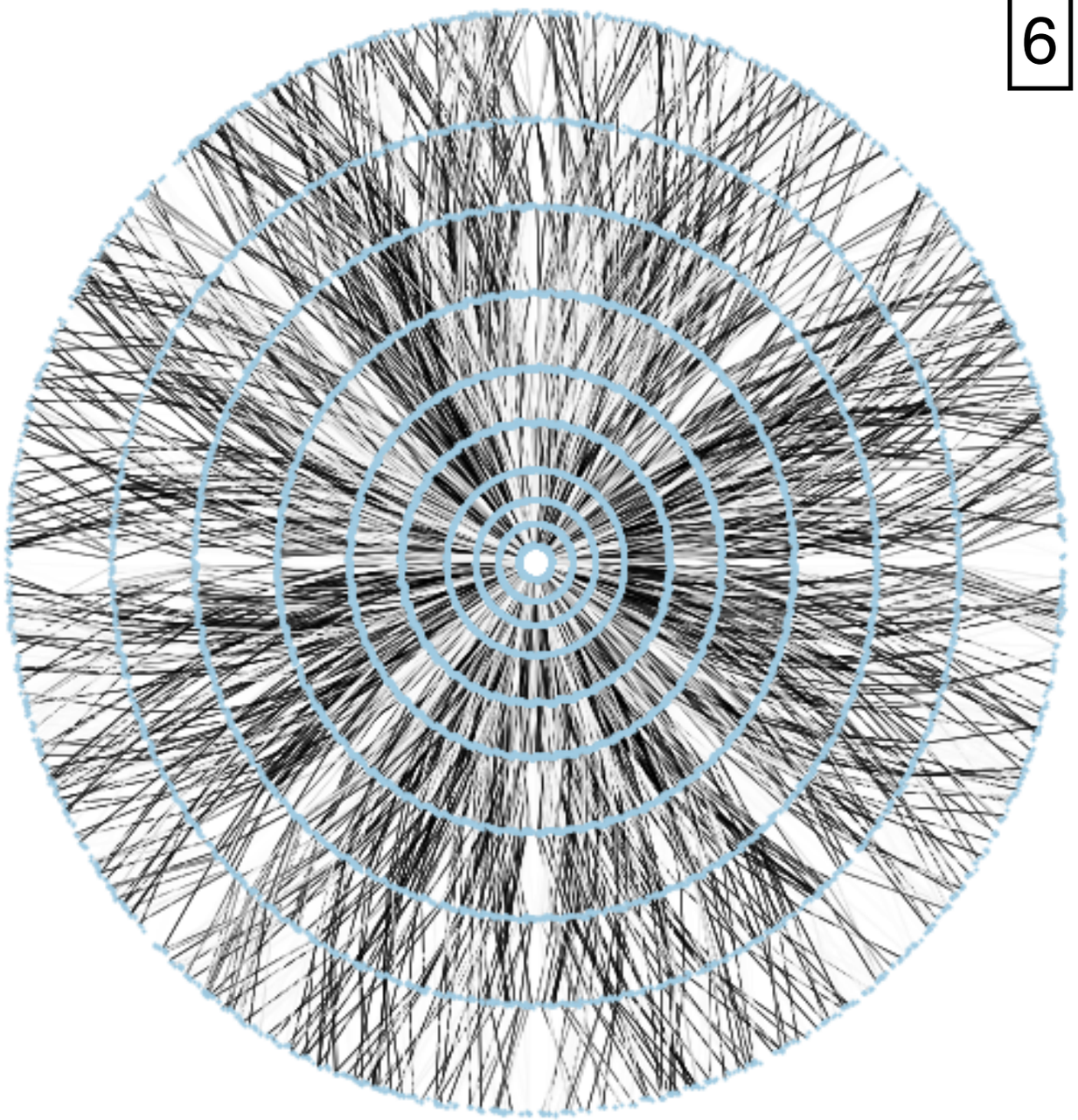


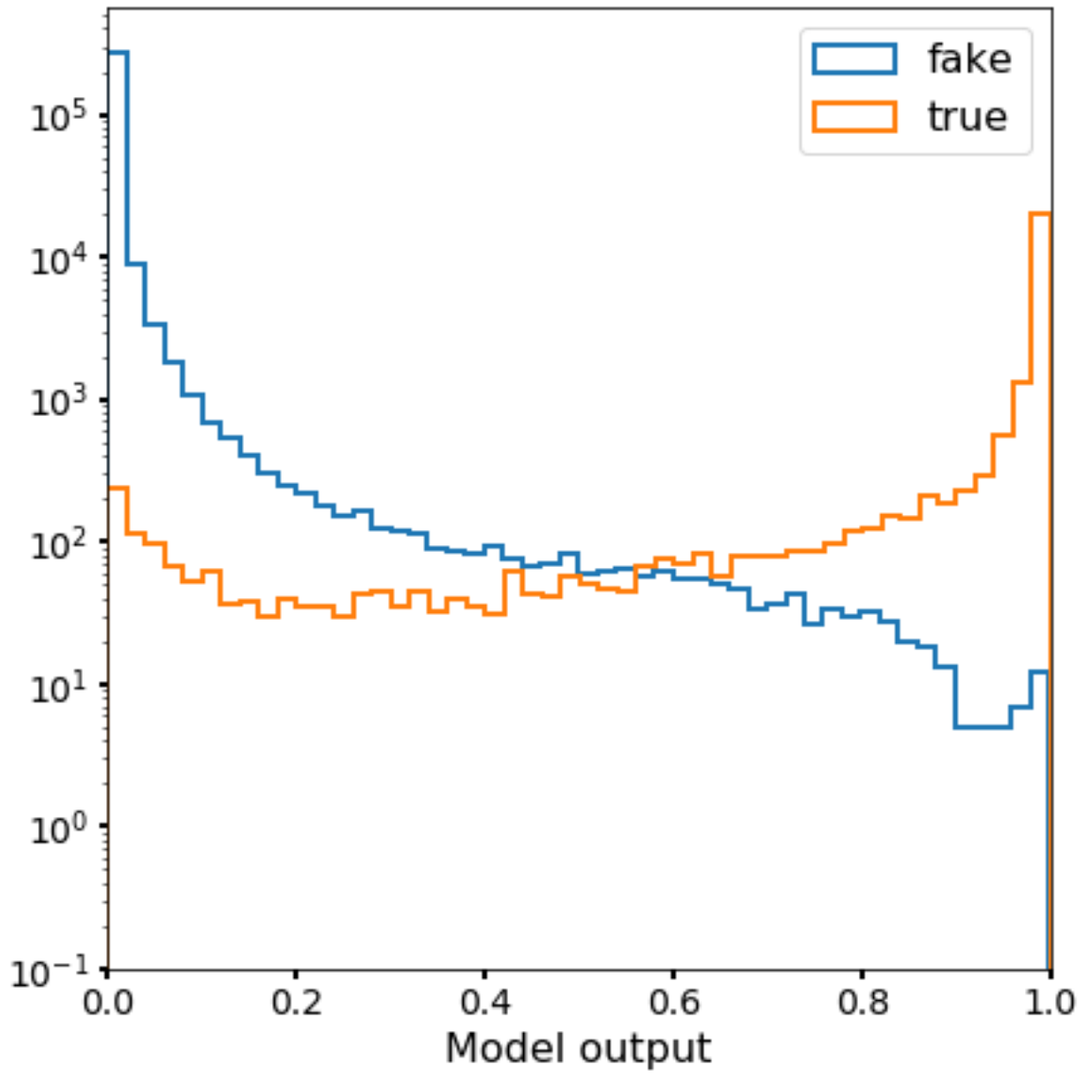


Edges with higher scores are darker than that with lower scores

Edges with scores < 0.01 are removed for visualization purpose.

Predicted score improves after each iteration

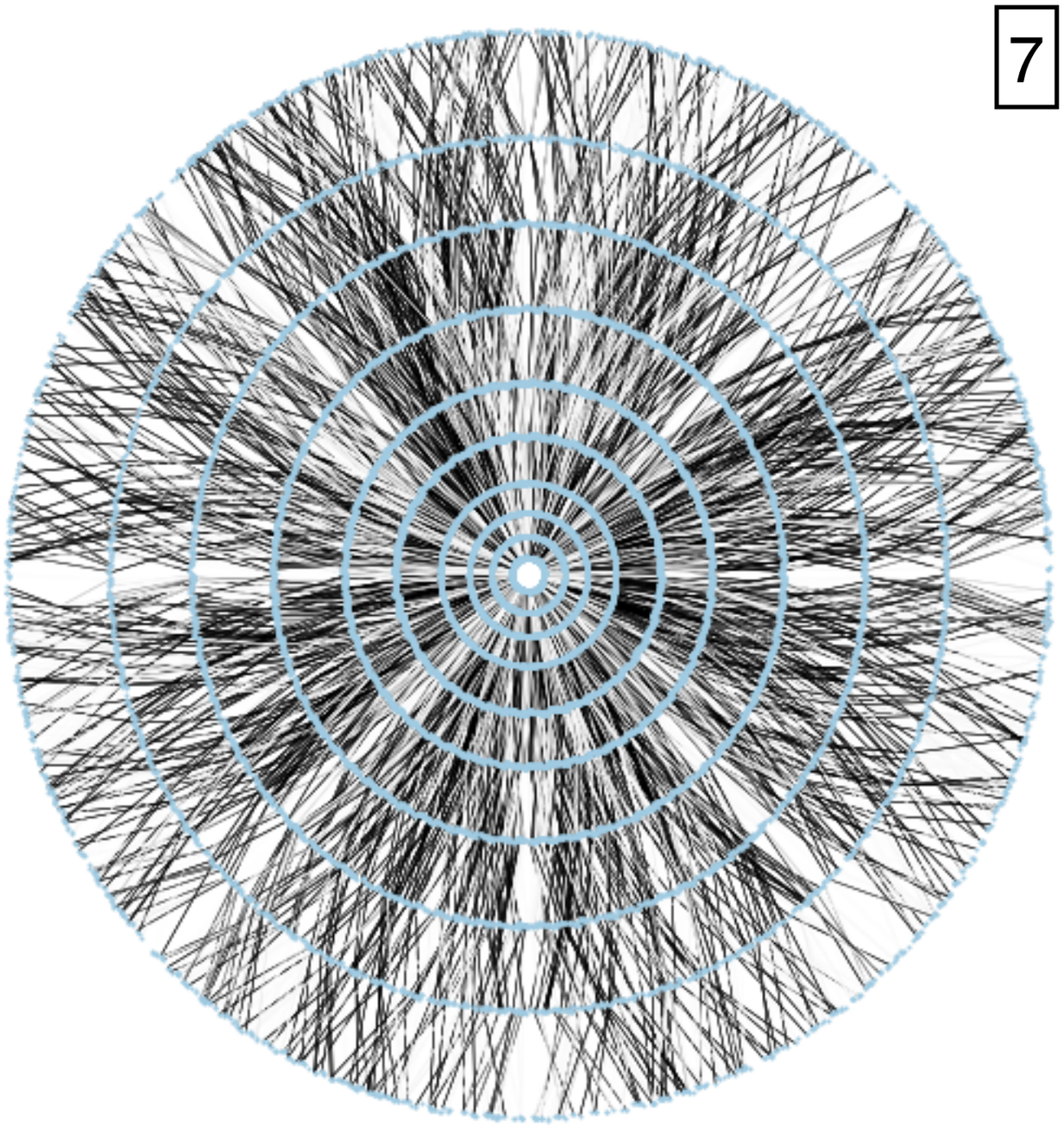


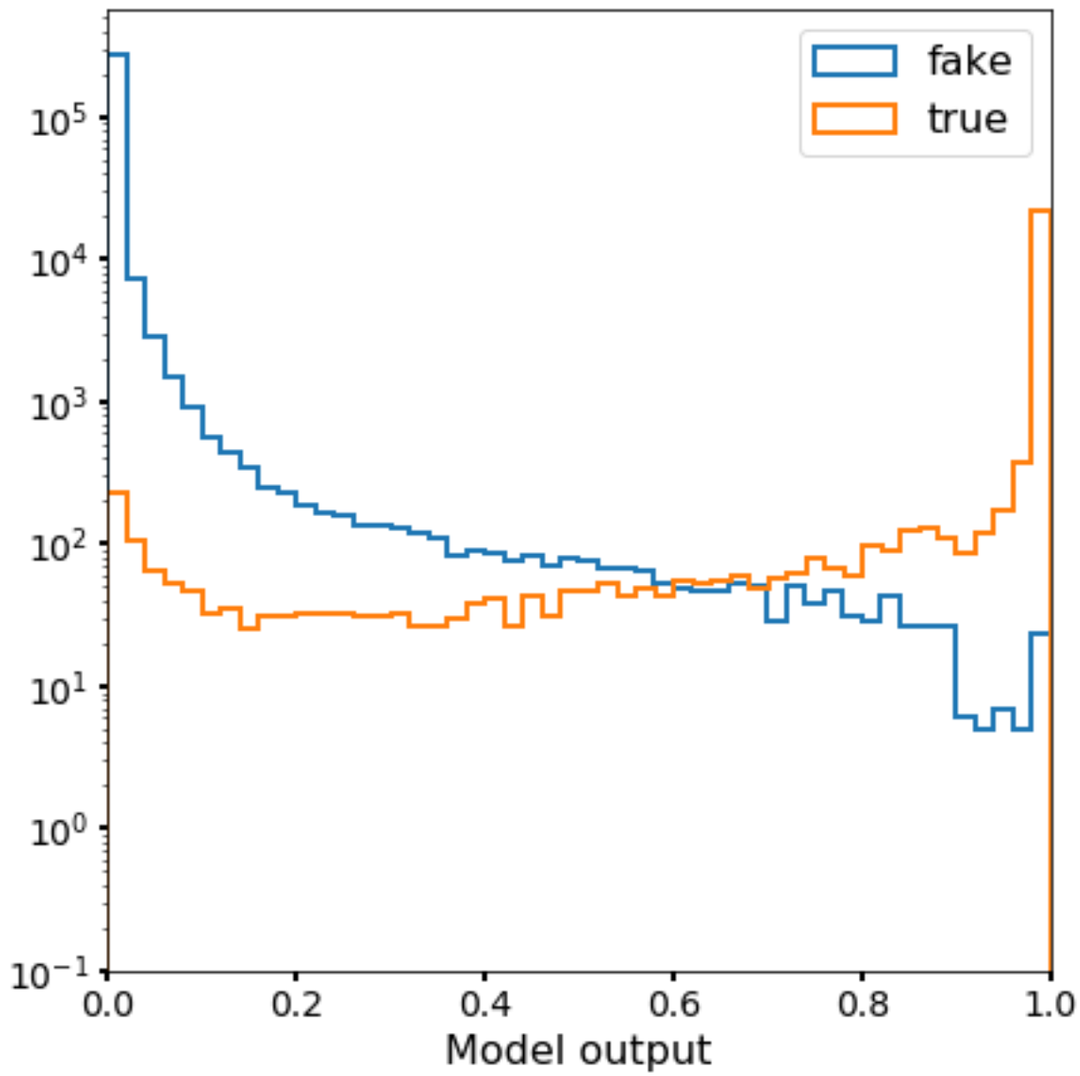


Edges with higher scores are darker than that with lower scores

Edges with scores < 0.01 are removed for visualization purpose.

Predicted score improves after each iteration

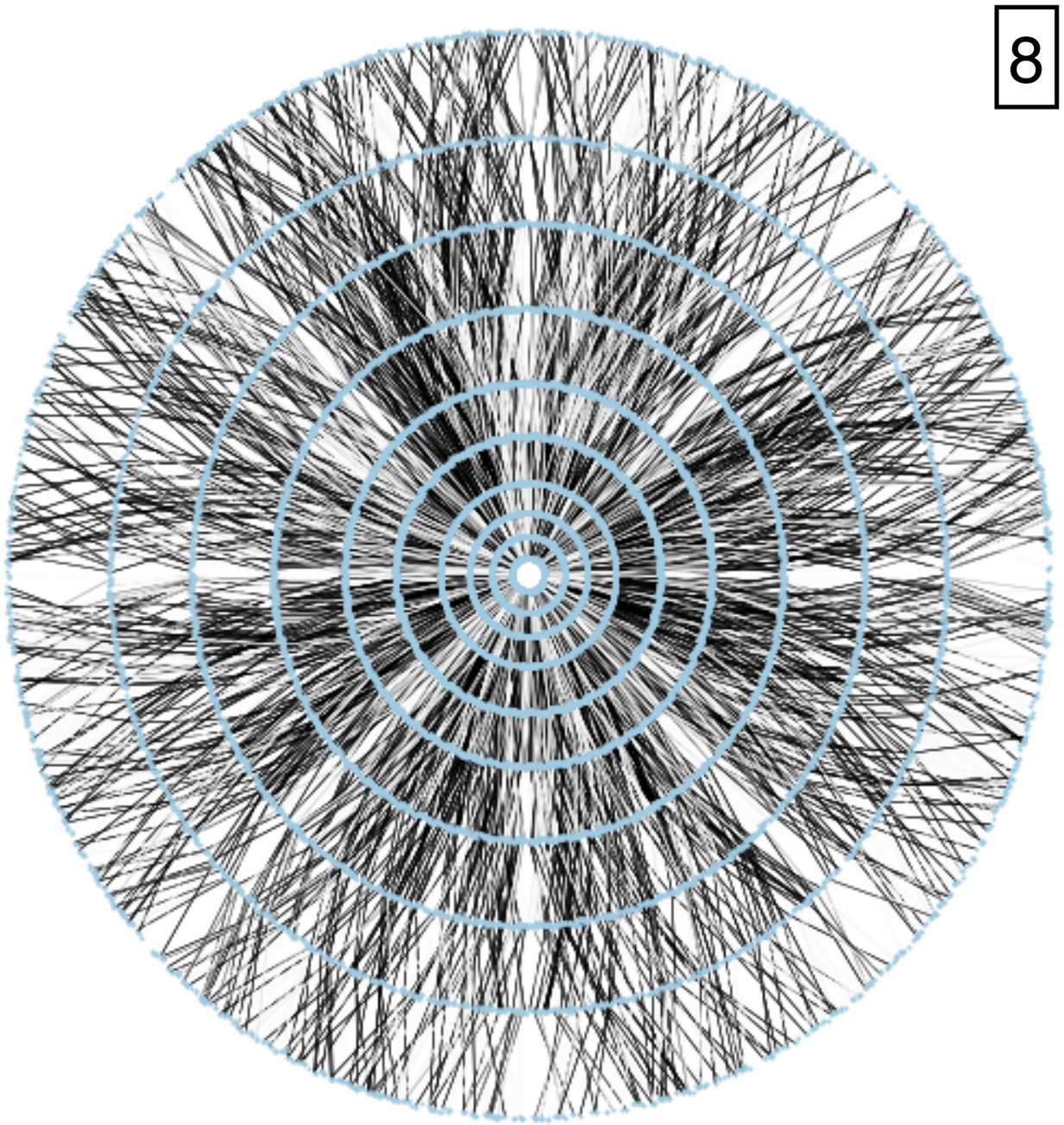


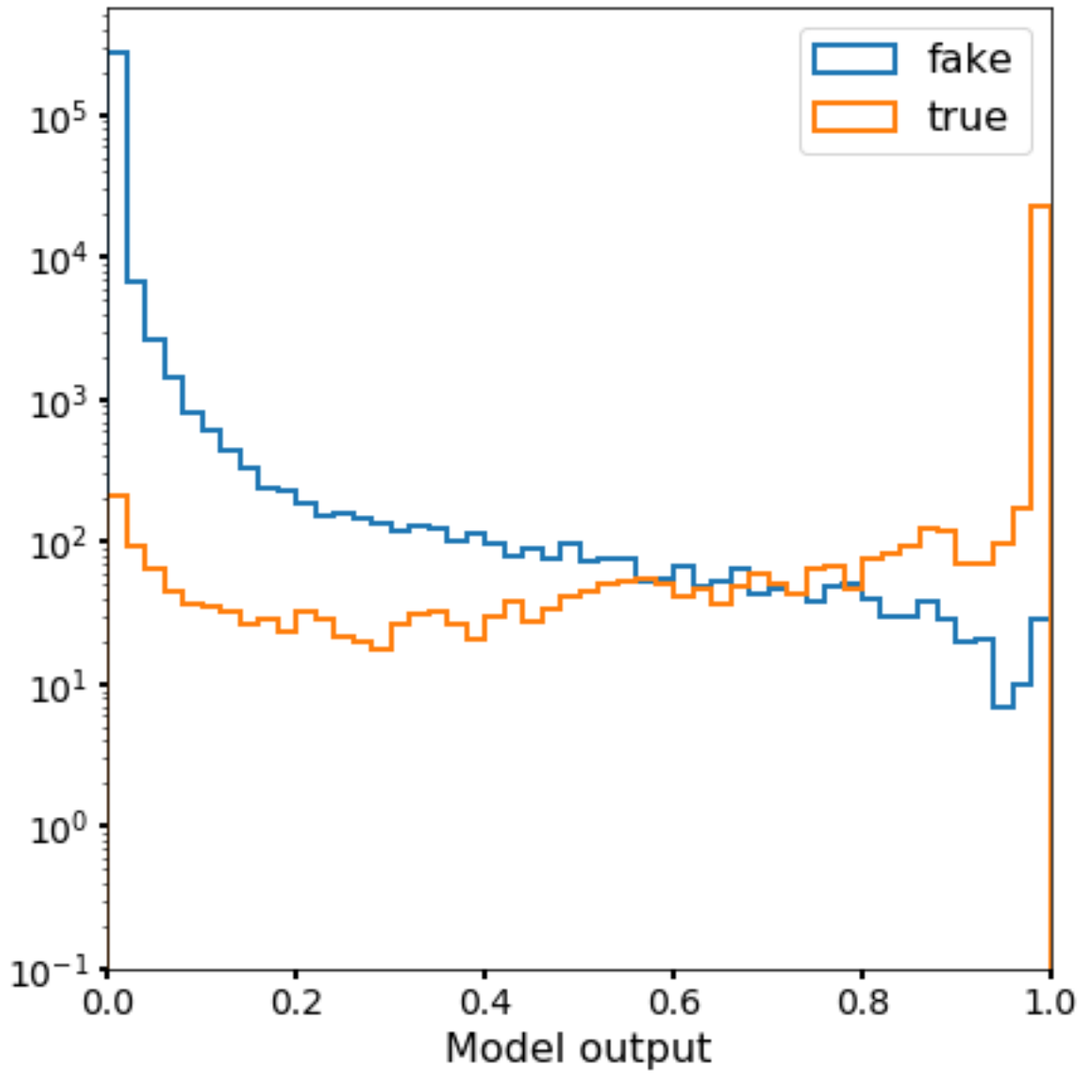


Edges with higher scores are darker than that with lower scores

Edges with scores < 0.01 are removed for visualization purpose.

Predicted score improves after each iteration



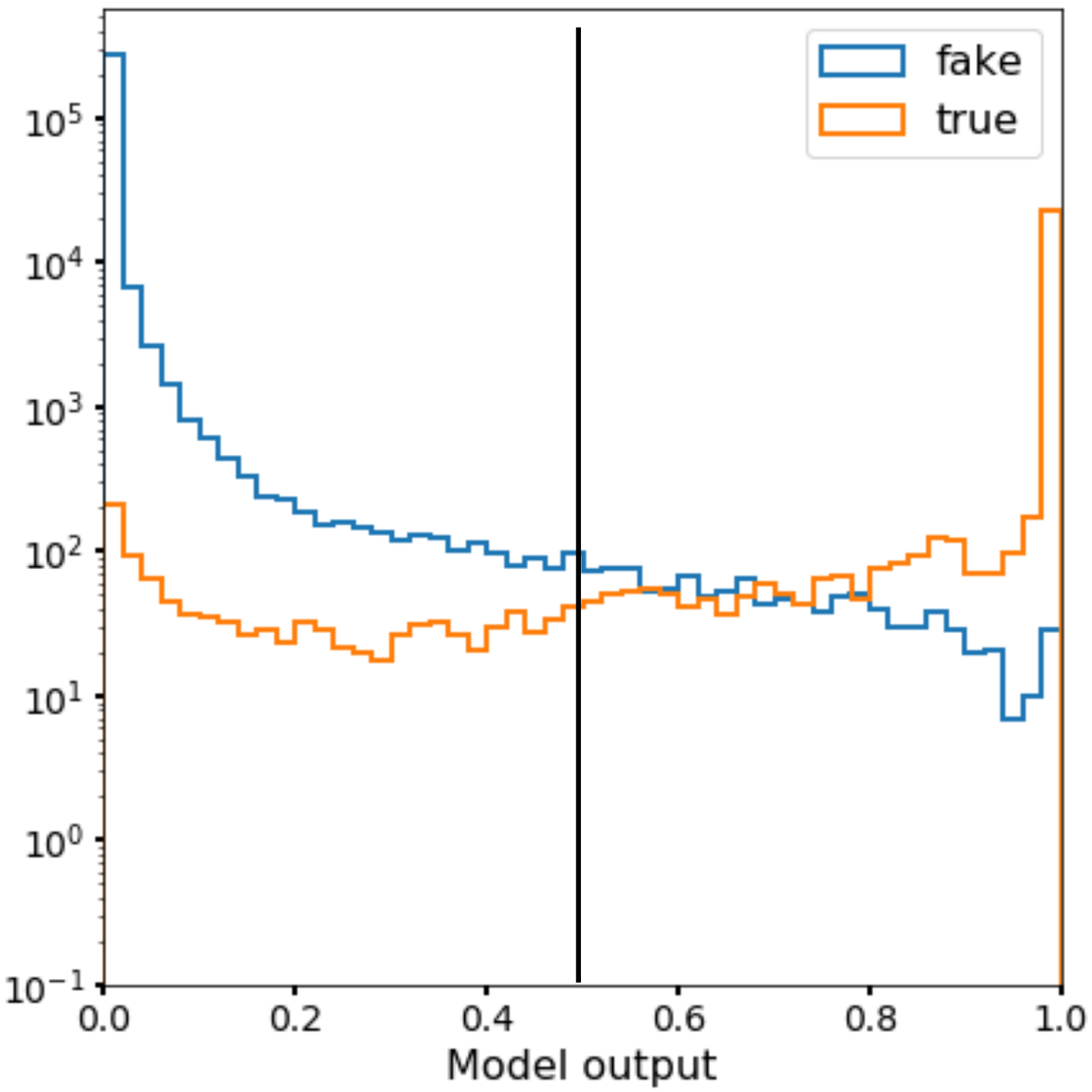


Edges with higher scores are darker than that with lower scores

Edges with scores < 0.01 are removed for visualization purpose.

Performances

- ~266k tunable parameters in TensorFlow
- Trained on a GPU for about 2 epochs
- Weighted loss function

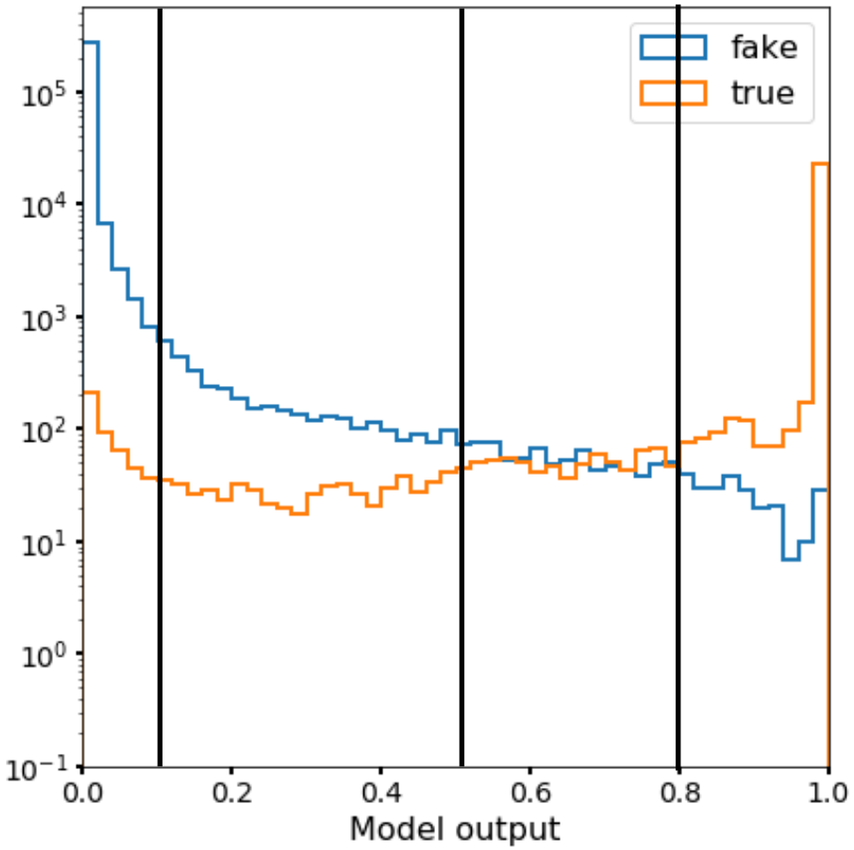


Threshold	0.1	0.5	0.8
Edge Efficiency	98.2%	95.9%	93.0%
Edge Purity	84.0%	95.7%	98.9%

$$\text{Efficiency} = \frac{\#\text{ of True Edges passed the threshold}}{\#\text{ of total True Edges}}$$

$$\text{Purity} = \frac{\#\text{ of True Edges passed the threshold}}{\#\text{ of total Edges passed the threshold}}$$

Connect The Dots, a simple algorithm



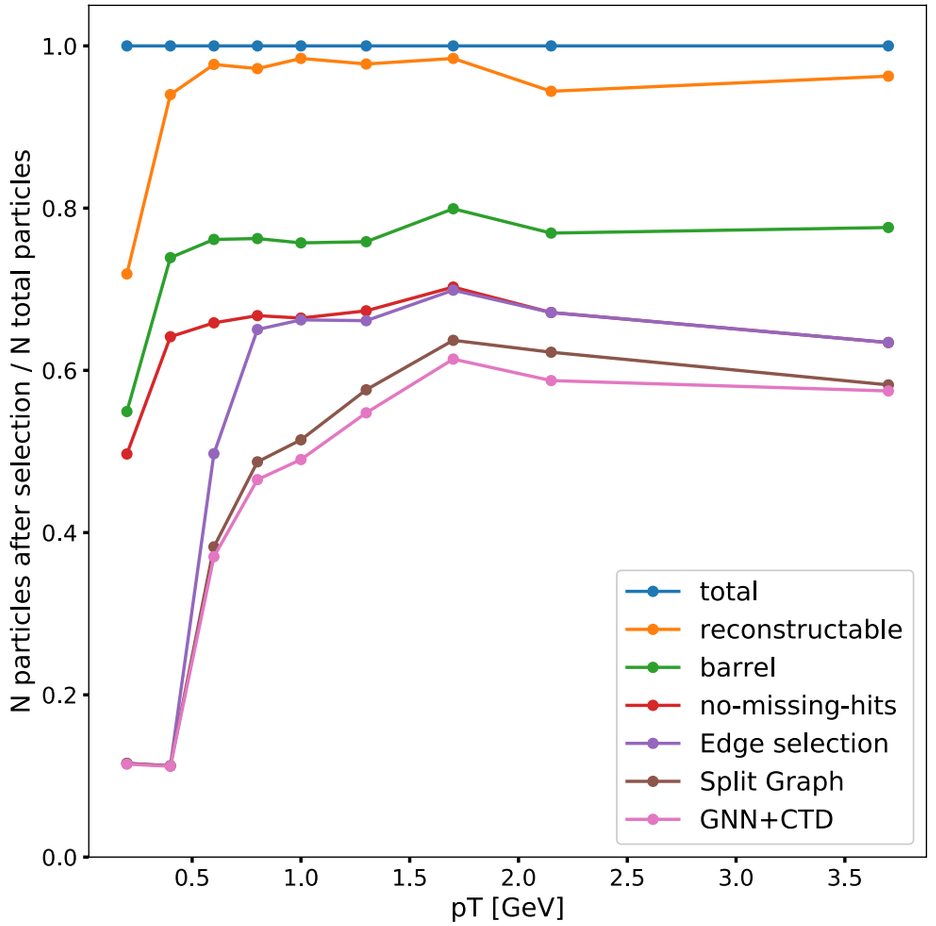
Guided by edge scores from GNN, we walk through the graph from inside to outside along edges with the maximum score that is **> 0.1,** as ones < 0.1 having high probability being fake

Add paths with scores **> 0.8** → having high probability being true

- Longest path is selected for the starting hit, then go to next not-used hit.
- Each hit is assigned to one track.
  - We will lift the constraint to gain efficiency and robustness, and then resolve ambiguities.

A summary

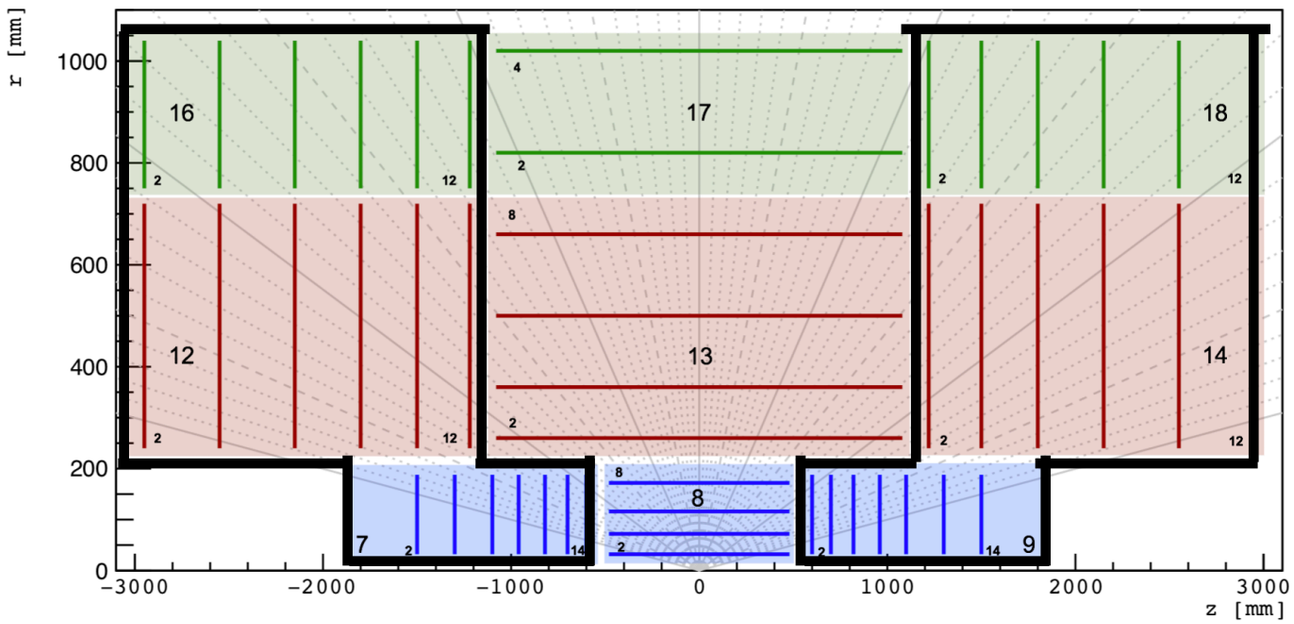
one-event	N-particles	ratio w.r.t Total	ratio w.r.t Reconstructable	relative ratio
Total	11170	100%		100%
Reconstructable	9635	86%	100%	86%
Barrel	7492	67%	78%	78%
No-missing hits	6600	59%	69%	88%
Edge selection	3114	28%	32%	**47%**
Split graph	2668	24%	28%	**86%**
GNN	2590	23%	27%	**97%**



GNN edge classifier achieves over 95% efficiency across the pT range with a purity greater than 95%

Edge selections

- Current selections tuned to be efficiency for tracks with pT > 1 GeV in barrel.
  - Not tested in endcap (probability need tuning)
- Explore simple neutral networks to do the job:
  - No need of engineering variables
  - Applicable to all hits pairing (barrel and endcap)
  - Can easily use GPU and be parallelized.



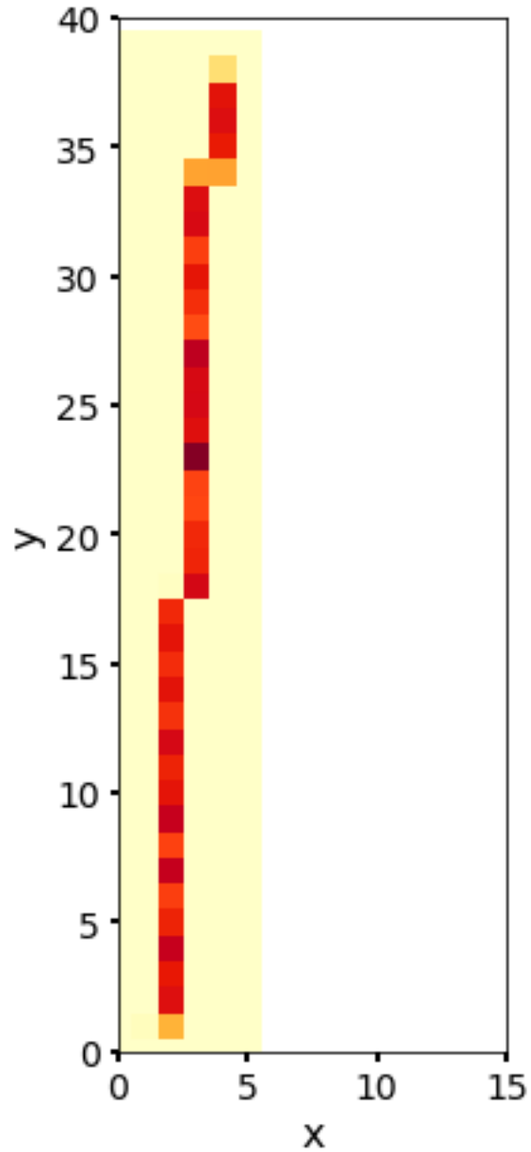
Build 90 promising layer-pairs,

each having a Neutral Network to select the right pairs.

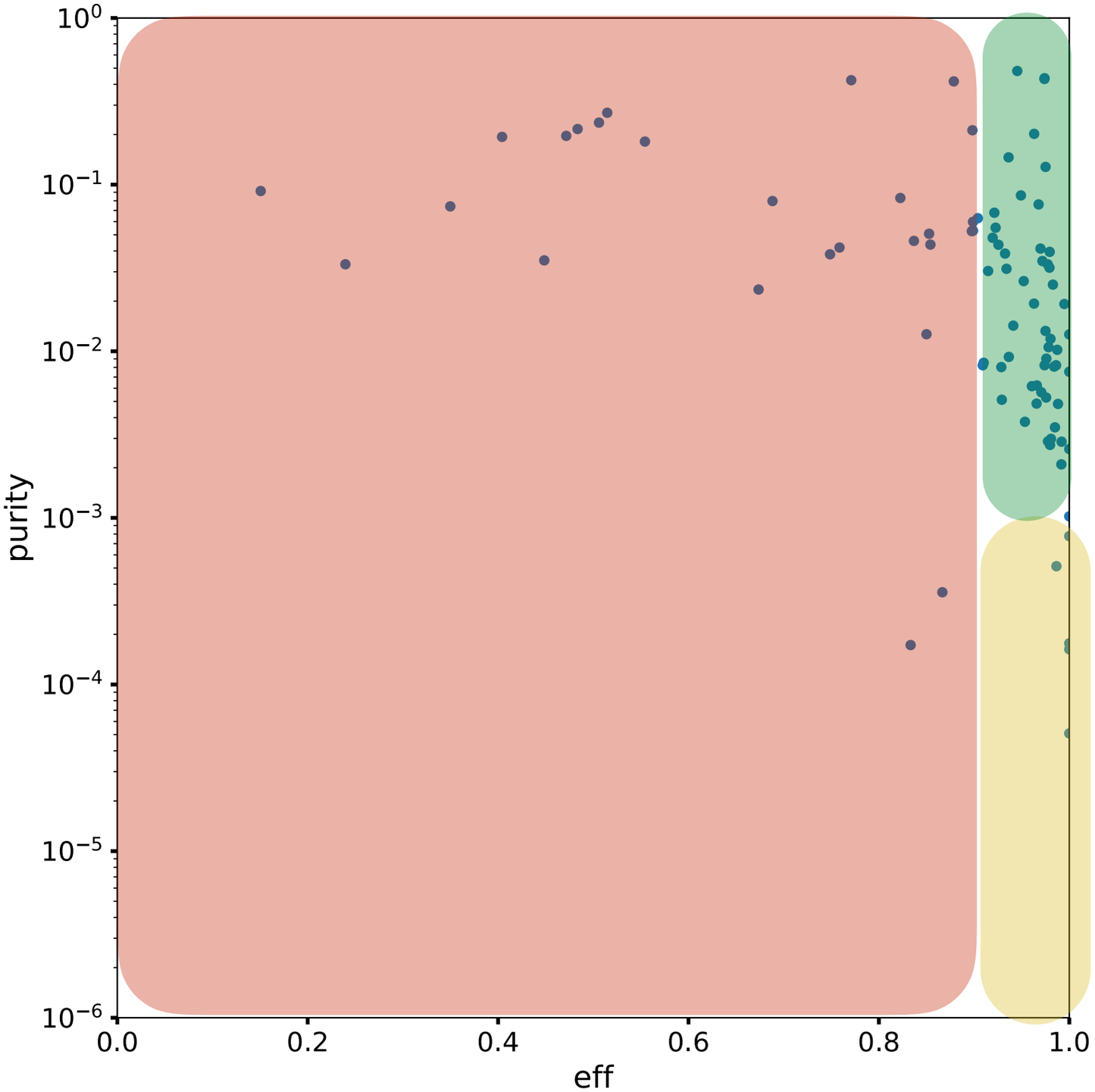
Neural Network input variables

- Hit Location variables: x, y, z
- Cluster Info: x, y, z
  - x-y-z on the module:
    - x = (max(ch0) - min(ch0)) *pitch_x;
    - y = (max(ch1) - min(ch1)) * pitch_y;
    - z = module_width
  - local x-y-z —> global x-y-z
    - Use rotation-matrix and translation vector
- **In total 12 input variables**.

Noise hits and duplicated hits are used as well.



Neural network performance



Most Lay-pairs achieve > 90% efficiency at the cost of purity ~0.1%.

Can perform fine-tuning for these handful worse-performed NNs

Will proceed to construct the graph for all events.

Short-term focus

1. Complete full event process (including endcap and noise hits) using the NNs + GNN + CTD.
2. Deal with unknown missing hits and duplicated hits
    1. Join the efforts from traditional algorithms
3. Scale GNN model:
    1. GNN normally does not have good scaleability;
    2. Active research efforts from machine learning